

## Linux Software (DM7820/DM9820)

Generated by Doxygen 1.8.1

Wed Jul 8 2015 09:17:58



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	DM7820 driver header file . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.2	DM7820 driver enumerations . . . . .	8
4.2.1	Detailed Description . . . . .	8
4.2.2	Enumeration Type Documentation . . . . .	8
4.2.2.1	dm7820_pci_region_access_dir . . . . .	8
4.3	DM7820 driver macros . . . . .	9
4.3.1	Detailed Description . . . . .	9
4.3.2	Macro Definition Documentation . . . . .	9
4.3.2.1	DM7820_MAX_DMA_BUFFER_COUNT . . . . .	9
4.3.2.2	DM7820_MAX_DMA_BUFFER_SIZE . . . . .	9
4.4	DM7820 driver constants . . . . .	10
4.4.1	Detailed Description . . . . .	10
4.5	DM7820 driver structures . . . . .	11
4.5.1	Detailed Description . . . . .	11
4.6	DM7820 driver forward declarations . . . . .	12
4.6.1	Detailed Description . . . . .	12
4.7	DM7820 driver functions . . . . .	13
4.7.1	Detailed Description . . . . .	14
4.7.2	Function Documentation . . . . .	15
4.7.2.1	dm7820_access_pci_region . . . . .	15
4.7.2.2	dm7820_allocate_irq . . . . .	15

4.7.2.3	<a href="#">dm7820_dequeue_interrupt</a>	15
4.7.2.4	<a href="#">dm7820_disable_all_interrupts</a>	16
4.7.2.5	<a href="#">dm7820_dma_check_xfer</a>	16
4.7.2.6	<a href="#">dm7820_dma_pause</a>	16
4.7.2.7	<a href="#">dm7820_dma_read</a>	16
4.7.2.8	<a href="#">dm7820_dma_stop</a>	17
4.7.2.9	<a href="#">dm7820_dma_write</a>	17
4.7.2.10	<a href="#">dm7820_enable_plx_interrupts</a>	17
4.7.2.11	<a href="#">dm7820_free_dma_mappings</a>	18
4.7.2.12	<a href="#">dm7820_get_buffer_phy_addr</a>	18
4.7.2.13	<a href="#">dm7820_get_interrupt_status</a>	18
4.7.2.14	<a href="#">dm7820_get_pci_master_status</a>	19
4.7.2.15	<a href="#">dm7820_initialize_device_descriptor</a>	19
4.7.2.16	<a href="#">dm7820_initialize_dma</a>	19
4.7.2.17	<a href="#">dm7820_initialize_hardware</a>	20
4.7.2.18	<a href="#">dm7820_int_queue_add</a>	20
4.7.2.19	<a href="#">dm7820_interrupt_handler</a>	20
4.7.2.20	<a href="#">dm7820_ioctl</a>	21
4.7.2.21	<a href="#">dm7820_load</a>	21
4.7.2.22	<a href="#">dm7820_modify_pci_region</a>	22
4.7.2.23	<a href="#">dm7820_open</a>	22
4.7.2.24	<a href="#">dm7820_poll</a>	22
4.7.2.25	<a href="#">dm7820_probe_device_blocks</a>	23
4.7.2.26	<a href="#">dm7820_probe_devices</a>	24
4.7.2.27	<a href="#">dm7820_process_pci_regions</a>	24
4.7.2.28	<a href="#">dm7820_read_pci_region</a>	25
4.7.2.29	<a href="#">dm7820_register_char_device</a>	25
4.7.2.30	<a href="#">dm7820_release</a>	26
4.7.2.31	<a href="#">dm7820_release_resources</a>	26
4.7.2.32	<a href="#">dm7820_service_dma_function</a>	26
4.7.2.33	<a href="#">dm7820_unregister_char_device</a>	27
4.7.2.34	<a href="#">dm7820_validate_device</a>	27
4.7.2.35	<a href="#">dm7820_validate_pci_access</a>	28
4.7.2.36	<a href="#">dm7820_write_pci_region</a>	28
4.8	<a href="#">DM7820 global variable header file</a>	30
4.8.1	<a href="#">Detailed Description</a>	30
4.9	<a href="#">DM7820 ioctl header file</a>	31
4.9.1	<a href="#">Detailed Description</a>	31
4.10	<a href="#">DM7820 ioctl enumerations</a>	32
4.10.1	<a href="#">Detailed Description</a>	32

4.10.2 Enumeration Type Documentation . . . . .	32
4.10.2.1 dm7820_dma_manage_function . . . . .	32
4.11 DM7820 ioctl structures . . . . .	33
4.11.1 Detailed Description . . . . .	33
4.11.2 Typedef Documentation . . . . .	33
4.11.2.1 dm7820_ioctl_region_readwrite_t . . . . .	33
4.12 DM7820 ioctl macros . . . . .	34
4.12.1 Detailed Description . . . . .	34
4.13 DM7820 user library header file . . . . .	35
4.13.1 Detailed Description . . . . .	35
4.14 DM7820 user library macros . . . . .	36
4.14.1 Detailed Description . . . . .	36
4.14.2 Macro Definition Documentation . . . . .	36
4.14.2.1 DM7820_DMA_DEMAND_OFF_DM7820_TO_PCI . . . . .	36
4.14.2.2 DM7820_DMA_DEMAND_OFF_PCI_TO_DM7820 . . . . .	36
4.14.2.3 DM7820_DMA_DEMAND_ON_DM7820_TO_PCI . . . . .	36
4.14.2.4 DM7820_DMA_DEMAND_ON_PCI_TO_DM7820 . . . . .	36
4.14.2.5 DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION . . . . .	37
4.14.2.6 DM7820_INCENC_ENABLE_PHASE_FILTER_TRANSITION . . . . .	37
4.14.2.7 DM7820_INCENC_RESET_PHASE_FILTER . . . . .	37
4.14.2.8 DM7820_INTERRUPT_STATUS_IS_SOURCE_PENDING . . . . .	37
4.14.2.9 DM7820_Return_Status . . . . .	38
4.15 DM7820 user library type definitions . . . . .	39
4.15.1 Detailed Description . . . . .	39
4.16 DM7820 user library structures . . . . .	40
4.16.1 Detailed Description . . . . .	40
4.16.2 Typedef Documentation . . . . .	40
4.16.2.1 DM7820_Board_Descriptor . . . . .	40
4.17 DM7820 user library functions . . . . .	41
4.17.1 Detailed Description . . . . .	41
4.18 DM7820 user library advanced interrupt functions . . . . .	42
4.18.1 Detailed Description . . . . .	42
4.18.2 Function Documentation . . . . .	42
4.18.2.1 DM7820_AdvInt_Get_Status . . . . .	42
4.18.2.2 DM7820_AdvInt_Read_Capture . . . . .	43
4.18.2.3 DM7820_AdvInt_Set_Compare . . . . .	43
4.18.2.4 DM7820_AdvInt_Set_Mask . . . . .	44
4.18.2.5 DM7820_AdvInt_Set_Master . . . . .	44
4.18.2.6 DM7820_AdvInt_Set_Mode . . . . .	45
4.19 DM7820 user library FIFO functions . . . . .	46

4.19.1	Detailed Description	46
4.19.2	Function Documentation	47
4.19.2.1	DM7820_FIFO_DMA_Configure	47
4.19.2.2	DM7820_FIFO_DMA_Create_Buffer	47
4.19.2.3	DM7820_FIFO_DMA_Enable	48
4.19.2.4	DM7820_FIFO_DMA_Free_Buffer	48
4.19.2.5	DM7820_FIFO_DMA_Initialize	48
4.19.2.6	DM7820_FIFO_DMA_Read	49
4.19.2.7	DM7820_FIFO_DMA_Write	50
4.19.2.8	DM7820_FIFO_Enable	50
4.19.2.9	DM7820_FIFO_Get_Status	51
4.19.2.10	DM7820_FIFO_Read	51
4.19.2.11	DM7820_FIFO_Set_Data_Input	52
4.19.2.12	DM7820_FIFO_Set_DMA_Request	52
4.19.2.13	DM7820_FIFO_Set_Input_Clock	53
4.19.2.14	DM7820_FIFO_Set_Output_Clock	53
4.19.2.15	DM7820_FIFO_Write	54
4.19.2.16	DM7820_Stop_DMA	54
4.20	DM7820 user library general functions	55
4.20.1	Detailed Description	55
4.20.2	Function Documentation	55
4.20.2.1	DM7820_General_Close_Board	55
4.20.2.2	DM7820_General_Enable_Interrupt	56
4.20.2.3	DM7820_General_Get_Interrupt_Status	56
4.20.2.4	DM7820_General_Get_Version_Info	57
4.20.2.5	DM7820_General_Is_PCI_Master	58
4.20.2.6	DM7820_General_Open_Board	58
4.20.2.7	DM7820_General_RemoveISR	58
4.20.2.8	DM7820_General_Reset	59
4.20.2.9	DM7820_General_SetISRPriority	59
4.21	DM7820 user library incremental encoder functions	60
4.21.1	Detailed Description	60
4.21.2	Function Documentation	60
4.21.2.1	DM7820_IncEnc_Configure	60
4.21.2.2	DM7820_IncEnc_Enable	61
4.21.2.3	DM7820_IncEnc_Enable_Hold	62
4.21.2.4	DM7820_IncEnc_Get_Independent_Value	62
4.21.2.5	DM7820_IncEnc_Get_Joined_Value	62
4.21.2.6	DM7820_IncEnc_Get_Status	63
4.21.2.7	DM7820_IncEnc_Set_Independent_Value	64

4.21.2.8	DM7820_IncEnc_Set_Joined_Value . . . . .	64
4.21.2.9	DM7820_IncEnc_Set_Master . . . . .	64
4.22	DM7820 user library pulse width modulator functions . . . . .	66
4.22.1	Detailed Description . . . . .	66
4.22.2	Function Documentation . . . . .	66
4.22.2.1	DM7820_PWM_Enable . . . . .	66
4.22.2.2	DM7820_PWM_Set_Period . . . . .	66
4.22.2.3	DM7820_PWM_Set_Period_Master . . . . .	67
4.22.2.4	DM7820_PWM_Set_Width . . . . .	68
4.22.2.5	DM7820_PWM_Set_Width_Master . . . . .	68
4.23	DM7820 user library programmable clock functions . . . . .	70
4.23.1	Detailed Description . . . . .	70
4.23.2	Function Documentation . . . . .	70
4.23.2.1	DM7820_PrgClk_Set_Master . . . . .	70
4.23.2.2	DM7820_PrgClk_Set_Mode . . . . .	71
4.23.2.3	DM7820_PrgClk_Set_Period . . . . .	71
4.23.2.4	DM7820_PrgClk_Set_Start_Trigger . . . . .	72
4.23.2.5	DM7820_PrgClk_Set_Stop_Trigger . . . . .	72
4.24	DM7820 user library standard I/O functions . . . . .	73
4.24.1	Detailed Description . . . . .	73
4.24.2	Function Documentation . . . . .	73
4.24.2.1	DM7820_StdIO_Get_Input . . . . .	73
4.24.2.2	DM7820_StdIO_Set_IO_Mode . . . . .	74
4.24.2.3	DM7820_StdIO_Set_Output . . . . .	74
4.24.2.4	DM7820_StdIO_Set_Periph_Mode . . . . .	75
4.24.2.5	DM7820_StdIO_Strobe_Input . . . . .	75
4.24.2.6	DM7820_StdIO_Strobe_Mode . . . . .	75
4.24.2.7	DM7820_StdIO_Strobe_Output . . . . .	76
4.25	DM7820 user library 8254 timer/counter functions . . . . .	77
4.25.1	Detailed Description . . . . .	77
4.25.2	Function Documentation . . . . .	77
4.25.2.1	DM7820_TmrCtr_Get_Status . . . . .	77
4.25.2.2	DM7820_TmrCtr_Program . . . . .	78
4.25.2.3	DM7820_TmrCtr_Read . . . . .	78
4.25.2.4	DM7820_TmrCtr_Select_Clock . . . . .	79
4.25.2.5	DM7820_TmrCtr_Select_Gate . . . . .	79
4.26	DM7820 register header file . . . . .	80
4.26.1	Detailed Description . . . . .	80
4.27	DM7820 register BAR0 (memory-mapped PLX registers) offsets . . . . .	81
4.27.1	Detailed Description . . . . .	81

4.28 DM7820 register BAR1 (I/O-mapped PLX registers) offsets . . . . .	82
4.28.1 Detailed Description . . . . .	82
4.29 DM7820 register BAR2 (memory-mapped FPGA registers) offsets . . . . .	83
4.29.1 Detailed Description . . . . .	83
4.29.2 Macro Definition Documentation . . . . .	84
4.29.2.1 PLX9056_DMA_BURST . . . . .	84
4.29.2.2 PLX9056_DMA_CLEAR_COUNT . . . . .	84
4.29.2.3 PLX9056_DMA_DAC_CHAIN . . . . .	84
4.29.2.4 PLX9056_DMA_DEMAND_MODE . . . . .	84
4.29.2.5 PLX9056_DMA_DONE_INTERRUPT . . . . .	84
4.29.2.6 PLX9056_DMA_EOT_ENABLE . . . . .	84
4.29.2.7 PLX9056_DMA_EOT_END . . . . .	84
4.29.2.8 PLX9056_DMA_FAST_SLOW_TERM . . . . .	84
4.29.2.9 PLX9056_DMA_INTERRUPT_SEL . . . . .	85
4.29.2.10 PLX9056_DMA_LOCAL_ADDRESSING_MODE . . . . .	85
4.29.2.11 PLX9056_DMA_LOCAL_BURST . . . . .	85
4.29.2.12 PLX9056_DMA_MEMWRITE_INV . . . . .	85
4.29.2.13 PLX9056_DMA_READY . . . . .	85
4.29.2.14 PLX9056_DMA_RING_CONTROL . . . . .	85
4.29.2.15 PLX9056_DMA_RING_MODE . . . . .	85
4.29.2.16 PLX9056_DMA_SCATTERGATHER . . . . .	85
4.29.2.17 PLX9056_DMA_WAITSTATES_MASK . . . . .	86
4.29.2.18 PLX9056_DMA_WIDTH_16 . . . . .	86
4.29.2.19 PLX9056_DMA_WIDTH_32 . . . . .	86
4.29.2.20 PLX9056_DMA_WIDTH_8 . . . . .	86
4.29.2.21 PLX9056_DMA_WIDTH_MASK . . . . .	86
4.30 DM7820 register board control offsets . . . . .	87
4.30.1 Detailed Description . . . . .	87
4.31 DM7820 register standard I/O offsets . . . . .	88
4.31.1 Detailed Description . . . . .	88
4.32 DM7820 register 8254 timer/counter offsets . . . . .	89
4.32.1 Detailed Description . . . . .	89
4.33 DM7820 register FIFO 0 offsets . . . . .	90
4.33.1 Detailed Description . . . . .	90
4.34 DM7820 register FIFO 1 offsets . . . . .	91
4.34.1 Detailed Description . . . . .	91
4.35 DM7820 register programmable clock 0 offsets . . . . .	92
4.35.1 Detailed Description . . . . .	92
4.36 DM7820 register programmable clock 1 offsets . . . . .	93
4.36.1 Detailed Description . . . . .	93



4.37	DM7820 register programmable clock 2 offsets . . . . .	94
4.37.1	Detailed Description . . . . .	94
4.38	DM7820 register programmable clock 3 offsets . . . . .	95
4.38.1	Detailed Description . . . . .	95
4.39	DM7820 register advanced interrupt 0 offsets . . . . .	96
4.39.1	Detailed Description . . . . .	96
4.40	DM7820 register advanced interrupt 1 offsets . . . . .	97
4.40.1	Detailed Description . . . . .	97
4.41	DM7820 register incremental encoder 0 offsets . . . . .	98
4.41.1	Detailed Description . . . . .	98
4.42	DM7820 register incremental encoder 1 offsets . . . . .	99
4.42.1	Detailed Description . . . . .	99
4.43	DM7820 register pulse width modulator 0 offsets . . . . .	100
4.43.1	Detailed Description . . . . .	100
4.44	DM7820 register pulse width modulator 1 offsets . . . . .	101
4.44.1	Detailed Description . . . . .	101
4.45	DM7820 register 8254 timer/counter A offsets . . . . .	102
4.45.1	Detailed Description . . . . .	102
4.46	DM7820 register 8254 timer/counter B offsets . . . . .	103
4.46.1	Detailed Description . . . . .	103
4.47	DM7820 register PCI region lengths . . . . .	104
4.47.1	Detailed Description . . . . .	104
4.48	DM7820 register functional block identifiers . . . . .	105
4.48.1	Detailed Description . . . . .	105
4.49	DM7820 register functional block identifier values . . . . .	106
4.49.1	Detailed Description . . . . .	106
4.50	DM7820 register functional block identifier offsets . . . . .	107
4.50.1	Detailed Description . . . . .	107
4.51	DM7820 register BAR2 macros . . . . .	108
4.51.1	Detailed Description . . . . .	108
4.52	DM7820 register BAR2 FPGA Version Register . . . . .	109
4.52.1	Detailed Description . . . . .	109
4.53	DM7820 register BAR2 Board Reset Register . . . . .	110
4.53.1	Detailed Description . . . . .	110
4.54	DM7820 type definition header file . . . . .	111
4.54.1	Detailed Description . . . . .	111
4.55	DM7820 type enumerations . . . . .	112
4.55.1	Detailed Description . . . . .	112
4.56	DM7820 type PCI enumerations . . . . .	113
4.56.1	Detailed Description . . . . .	113

4.56.2	Enumeration Type Documentation	113
4.56.2.1	dm7820_pci_region_access_size	113
4.56.2.2	dm7820_pci_region_num	113
4.57	DM7820 type standard I/O enumerations	114
4.57.1	Detailed Description	114
4.57.2	Enumeration Type Documentation	114
4.57.2.1	_DM7820_StdIO_IO_Mode	114
4.57.2.2	_DM7820_StdIO_Periph_Mode	114
4.57.2.3	_DM7820_StdIO_Port	115
4.57.2.4	_DM7820_StdIO_Strobe	115
4.58	DM7820 type timer/counter enumerations	116
4.58.1	Detailed Description	117
4.58.2	Enumeration Type Documentation	117
4.58.2.1	_dm7820_tmrctr_clock	117
4.58.2.2	_dm7820_tmrctr_count_mode	117
4.58.2.3	_dm7820_tmrctr_gate	118
4.58.2.4	_dm7820_tmrctr_timer	118
4.58.2.5	_dm7820_tmrctr_waveform	119
4.59	DM7820 type programmable clock enumerations	120
4.59.1	Detailed Description	121
4.59.2	Enumeration Type Documentation	121
4.59.2.1	_dm7820_prgclk_clock	121
4.59.2.2	_dm7820_prgclk_master_clock	121
4.59.2.3	_dm7820_prgclk_mode	122
4.59.2.4	_dm7820_prgclk_start_trigger	122
4.59.2.5	_dm7820_prgclk_stop_trigger	123
4.60	DM7820 type pulse width modulator enumerations	125
4.60.1	Detailed Description	125
4.60.2	Enumeration Type Documentation	126
4.60.2.1	_dm7820_pwm_output	126
4.60.2.2	_dm7820_pwm_period_master_clock	126
4.60.2.3	_dm7820_pwm_width_master_clock	126
4.61	DM7820 type incremental encoder enumerations	128
4.61.1	Detailed Description	129
4.61.2	Enumeration Type Documentation	129
4.61.2.1	_dm7820_incenc_channel	129
4.61.2.2	_dm7820_incenc_channel_mode	129
4.61.2.3	_dm7820_incenc_encoder	129
4.61.2.4	_dm7820_incenc_input_mode	129
4.61.2.5	_dm7820_incenc_master_clock	130

4.61.2.6	<a href="#">_dm7820_incenc_phase_transition</a>	130
4.61.2.7	<a href="#">_dm7820_incenc_status_condition</a>	131
4.62	<a href="#">DM7820 type FIFO enumerations</a>	132
4.62.1	<a href="#">Detailed Description</a>	133
4.62.2	<a href="#">Enumeration Type Documentation</a>	133
4.62.2.1	<a href="#">_dm7820_fifo_data_input</a>	133
4.62.2.2	<a href="#">_dm7820_fifo_dma_request</a>	134
4.62.2.3	<a href="#">_dm7820_fifo_input_clock</a>	134
4.62.2.4	<a href="#">_dm7820_fifo_output_clock</a>	135
4.62.2.5	<a href="#">_dm7820_fifo_queue</a>	135
4.62.2.6	<a href="#">_dm7820_fifo_status_condition</a>	136
4.63	<a href="#">DM7820 type advanced interrupt enumerations</a>	137
4.63.1	<a href="#">Detailed Description</a>	137
4.63.2	<a href="#">Enumeration Type Documentation</a>	137
4.63.2.1	<a href="#">_dm7820_advint_interrupt</a>	137
4.63.2.2	<a href="#">_dm7820_advint_master_clock</a>	138
4.63.2.3	<a href="#">_dm7820_advint_mode</a>	138
4.64	<a href="#">DM7820 type interrupt enumerations</a>	139
4.64.1	<a href="#">Detailed Description</a>	139
4.64.2	<a href="#">Enumeration Type Documentation</a>	140
4.64.2.1	<a href="#">_dm7820_interrupt_source</a>	140
4.64.2.2	<a href="#">_dm7820_minor_interrupt_register</a>	141
4.65	<a href="#">DM7820 type definition structures</a>	142
4.65.1	<a href="#">Detailed Description</a>	142
4.65.2	<a href="#">Typedef Documentation</a>	142
4.65.2.1	<a href="#">dm7820_pci_access_request_t</a>	142
4.66	<a href="#">DM7820 type definition typedefs</a>	143
4.66.1	<a href="#">Detailed Description</a>	143
<b>5</b>	<b><a href="#">Data Structure Documentation</a></b>	<b>145</b>
5.1	<a href="#">_dm7820_interrupt_info Struct Reference</a>	145
5.1.1	<a href="#">Detailed Description</a>	145
5.1.2	<a href="#">Field Documentation</a>	145
5.1.2.1	<a href="#">error</a>	145
5.1.2.2	<a href="#">int_missed</a>	145
5.1.2.3	<a href="#">int_remaining</a>	145
5.1.2.4	<a href="#">source</a>	146
5.2	<a href="#">DM7820_Board_Descriptor Struct Reference</a>	146
5.2.1	<a href="#">Detailed Description</a>	146
5.2.2	<a href="#">Field Documentation</a>	146

5.2.2.1	<a href="#">file_descriptor</a>	146
5.2.2.2	<a href="#">isr</a>	146
5.2.2.3	<a href="#">pid</a>	146
5.3	<a href="#">dm7820_device_descriptor Struct Reference</a>	147
5.3.1	<a href="#">Detailed Description</a>	147
5.3.2	<a href="#">Field Documentation</a>	147
5.3.2.1	<a href="#">device_lock</a>	147
5.3.2.2	<a href="#">device_name</a>	147
5.3.2.3	<a href="#">dma_buffers_post_transfer</a>	147
5.3.2.4	<a href="#">dma_buffers_pre_transfer</a>	148
5.3.2.5	<a href="#">dma_in_read_direction</a>	148
5.3.2.6	<a href="#">dma_initialized</a>	148
5.3.2.7	<a href="#">dma_size</a>	148
5.3.2.8	<a href="#">dma_wait_queue</a>	148
5.3.2.9	<a href="#">int_queue_count</a>	148
5.3.2.10	<a href="#">int_queue_in</a>	148
5.3.2.11	<a href="#">int_queue_missed</a>	148
5.3.2.12	<a href="#">int_queue_out</a>	148
5.3.2.13	<a href="#">int_source_status</a>	149
5.3.2.14	<a href="#">int_status</a>	149
5.3.2.15	<a href="#">int_wait_queue</a>	149
5.3.2.16	<a href="#">interrupt_occurred</a>	149
5.3.2.17	<a href="#">irq_number</a>	149
5.3.2.18	<a href="#">pci</a>	149
5.3.2.19	<a href="#">reference_count</a>	149
5.3.2.20	<a href="#">remove_isr_flag</a>	149
5.4	<a href="#">dm7820_dma_descriptor_t Struct Reference</a>	149
5.4.1	<a href="#">Detailed Description</a>	150
5.4.2	<a href="#">Field Documentation</a>	150
5.4.2.1	<a href="#">bus_address</a>	150
5.4.2.2	<a href="#">virtual_address</a>	150
5.5	<a href="#">dm7820_dma_function_arguments Union Reference</a>	150
5.5.1	<a href="#">Detailed Description</a>	150
5.5.2	<a href="#">Field Documentation</a>	150
5.5.2.1	<a href="#">dma_init</a>	150
5.6	<a href="#">dm7820_dma_initialize_arguments Struct Reference</a>	151
5.6.1	<a href="#">Detailed Description</a>	151
5.6.2	<a href="#">Field Documentation</a>	151
5.6.2.1	<a href="#">buffer_count</a>	151
5.6.2.2	<a href="#">buffer_size</a>	151

5.7	<a href="#">dm7820_dma_list_item_t Struct Reference</a>	151
5.7.1	<a href="#">Detailed Description</a>	152
5.7.2	<a href="#">Field Documentation</a>	152
5.7.2.1	<a href="#">dma_buffer</a>	152
5.7.2.2	<a href="#">list</a>	152
5.8	<a href="#">dm7820_interrupt_control Struct Reference</a>	152
5.8.1	<a href="#">Detailed Description</a>	152
5.8.2	<a href="#">Field Documentation</a>	152
5.8.2.1	<a href="#">int_enable_bit</a>	152
5.8.2.2	<a href="#">int_status_bit</a>	153
5.8.2.3	<a href="#">minor_enable</a>	153
5.8.2.4	<a href="#">minor_reg</a>	153
5.8.2.5	<a href="#">minor_status</a>	153
5.9	<a href="#">dm7820_interrupt_status_source Struct Reference</a>	153
5.9.1	<a href="#">Detailed Description</a>	153
5.9.2	<a href="#">Field Documentation</a>	153
5.9.2.1	<a href="#">minor_reg</a>	153
5.9.2.2	<a href="#">source</a>	154
5.9.2.3	<a href="#">source_table</a>	154
5.10	<a href="#">dm7820_ioctl_argument Union Reference</a>	154
5.10.1	<a href="#">Detailed Description</a>	154
5.10.2	<a href="#">Field Documentation</a>	154
5.10.2.1	<a href="#">dma_function</a>	154
5.10.2.2	<a href="#">int_status</a>	154
5.10.2.3	<a href="#">modify</a>	155
5.10.2.4	<a href="#">readwrite</a>	155
5.11	<a href="#">dm7820_ioctl_dma_function Struct Reference</a>	155
5.11.1	<a href="#">Detailed Description</a>	155
5.11.2	<a href="#">Field Documentation</a>	155
5.11.2.1	<a href="#">arguments</a>	155
5.11.2.2	<a href="#">buffer_address</a>	155
5.11.2.3	<a href="#">direction</a>	156
5.11.2.4	<a href="#">DMA_Transfer_Size</a>	156
5.11.2.5	<a href="#">fifo</a>	156
5.11.2.6	<a href="#">function</a>	156
5.11.2.7	<a href="#">transfer_size</a>	156
5.11.2.8	<a href="#">user_buffer</a>	156
5.12	<a href="#">dm7820_ioctl_interrupt_status Struct Reference</a>	156
5.12.1	<a href="#">Detailed Description</a>	157
5.12.2	<a href="#">Field Documentation</a>	157

5.12.2.1	int_source_info . . . . .	157
5.12.2.2	wait_for_interrupt . . . . .	157
5.13	dm7820_ioctl_region_modify Struct Reference . . . . .	157
5.13.1	Detailed Description . . . . .	157
5.13.2	Field Documentation . . . . .	157
5.13.2.1	access . . . . .	157
5.13.2.2	mask . . . . .	158
5.13.2.3	mask16 . . . . .	158
5.13.2.4	mask32 . . . . .	158
5.13.2.5	mask8 . . . . .	158
5.14	dm7820_ioctl_region_readwrite Struct Reference . . . . .	158
5.14.1	Detailed Description . . . . .	158
5.14.2	Field Documentation . . . . .	159
5.14.2.1	access . . . . .	159
5.15	dm7820_minor_int_reg_layout Struct Reference . . . . .	159
5.15.1	Detailed Description . . . . .	159
5.15.2	Field Documentation . . . . .	159
5.15.2.1	enable_mask . . . . .	159
5.15.2.2	offset . . . . .	159
5.15.2.3	reserved_mask . . . . .	159
5.15.2.4	status_mask . . . . .	160
5.16	dm7820_pci_access_request Struct Reference . . . . .	160
5.16.1	Detailed Description . . . . .	160
5.16.2	Field Documentation . . . . .	160
5.16.2.1	data . . . . .	160
5.16.2.2	data16 . . . . .	160
5.16.2.3	data32 . . . . .	161
5.16.2.4	data8 . . . . .	161
5.16.2.5	offset . . . . .	161
5.16.2.6	region . . . . .	161
5.16.2.7	size . . . . .	161
5.17	dm7820_pci_region Struct Reference . . . . .	161
5.17.1	Detailed Description . . . . .	162
5.17.2	Field Documentation . . . . .	162
5.17.2.1	allocated . . . . .	162
5.17.2.2	io_addr . . . . .	162
5.17.2.3	length . . . . .	162
5.17.2.4	phys_addr . . . . .	162
5.17.2.5	virt_addr . . . . .	162
5.18	register_read Struct Reference . . . . .	162

5.18.1 Detailed Description . . . . .	163
5.18.2 Field Documentation . . . . .	163
5.18.2.1 expected . . . . .	163
5.18.2.2 offset . . . . .	163
5.18.2.3 region . . . . .	163
5.18.2.4 size . . . . .	163
<b>6 File Documentation</b>	<b>165</b>
6.1 examples/advint_event_interrupt.c File Reference . . . . .	165
6.1.1 Detailed Description . . . . .	165
6.1.2 Function Documentation . . . . .	166
6.1.2.1 ISR . . . . .	166
6.1.2.2 main . . . . .	166
6.1.3 Variable Documentation . . . . .	167
6.1.3.1 advint1_interrupts . . . . .	167
6.1.3.2 program_name . . . . .	167
6.2 examples/advint_match_interrupt.c File Reference . . . . .	167
6.2.1 Detailed Description . . . . .	168
6.2.2 Macro Definition Documentation . . . . .	168
6.2.2.1 COMPARE_REGISTER_VALUE . . . . .	168
6.2.3 Function Documentation . . . . .	168
6.2.3.1 main . . . . .	168
6.2.4 Variable Documentation . . . . .	169
6.2.4.1 program_name . . . . .	169
6.3 examples/advint_status.c File Reference . . . . .	169
6.3.1 Detailed Description . . . . .	170
6.3.2 Macro Definition Documentation . . . . .	170
6.3.2.1 COMPARE_REGISTER_VALUE . . . . .	170
6.3.3 Function Documentation . . . . .	170
6.3.3.1 main . . . . .	170
6.3.4 Variable Documentation . . . . .	171
6.3.4.1 program_name . . . . .	171
6.4 examples/advint_strobe_interrupt.c File Reference . . . . .	171
6.4.1 Detailed Description . . . . .	171
6.4.2 Function Documentation . . . . .	172
6.4.2.1 main . . . . .	172
6.4.3 Variable Documentation . . . . .	173
6.4.3.1 program_name . . . . .	173
6.5 examples/basic_test.c File Reference . . . . .	173
6.5.1 Detailed Description . . . . .	174

6.5.2	Macro Definition Documentation	175
6.5.2.1	DM7820_MAX_DMA_BUFFER_COUNT	175
6.5.2.2	DM7820_MAX_DMA_BUFFER_SIZE	175
6.5.3	Typedef Documentation	175
6.5.3.1	initialization_state_t	175
6.5.3.2	register_read_t	175
6.5.4	Enumeration Type Documentation	175
6.5.4.1	initialization_state	175
6.5.5	Function Documentation	176
6.5.5.1	expect_failure_and_check	176
6.5.5.2	expect_success	176
6.5.5.3	main	176
6.5.6	Variable Documentation	177
6.5.6.1	access_sizes	177
6.5.6.2	bad_device_name	177
6.5.6.3	descriptors	177
6.5.6.4	device_count	177
6.5.6.5	init_state	177
6.5.6.6	program_name	178
6.5.6.7	region_names	178
6.5.6.8	region_sizes	178
6.5.6.9	register_reads	178
6.6	examples/brdctl_device_info.c File Reference	179
6.6.1	Detailed Description	179
6.6.2	Function Documentation	179
6.6.2.1	main	179
6.6.3	Variable Documentation	180
6.6.3.1	program_name	180
6.7	examples/dma_capture_buffers.c File Reference	180
6.7.1	Detailed Description	181
6.7.2	Macro Definition Documentation	181
6.7.2.1	BUF_NUM	181
6.7.2.2	BUF_SIZE	181
6.7.2.3	DMA_BUF_SIZE	181
6.7.2.4	SAMPLES	182
6.7.3	Function Documentation	182
6.7.3.1	ISR	182
6.7.3.2	main	182
6.7.4	Variable Documentation	182
6.7.4.1	board	182



6.7.4.2	interrupts	183
6.7.4.3	program_name	183
6.8	examples/dma_capture_file.c File Reference	183
6.8.1	Detailed Description	184
6.8.2	Macro Definition Documentation	184
6.8.2.1	BUF_NUM	184
6.8.2.2	BUF_SIZE	184
6.8.2.3	DAT_FILE	185
6.8.2.4	DMA_BUF_SIZE	185
6.8.2.5	SAMPLES	185
6.8.3	Function Documentation	185
6.8.3.1	ISR	185
6.8.3.2	main	185
6.8.4	Variable Documentation	186
6.8.4.1	board	186
6.8.4.2	dma_done_interrupts	186
6.8.4.3	program_name	186
6.8.4.4	test_data	186
6.9	examples/fifo_cascaded.c File Reference	186
6.9.1	Detailed Description	187
6.9.2	Function Documentation	188
6.9.2.1	get_fifo_status	188
6.9.2.2	main	188
6.9.3	Variable Documentation	189
6.9.3.1	program_name	189
6.10	examples/fifo_interrupt.c File Reference	189
6.10.1	Detailed Description	189
6.10.2	Function Documentation	190
6.10.2.1	main	190
6.10.3	Variable Documentation	191
6.10.3.1	program_name	191
6.11	examples/fifo_pci_access.c File Reference	191
6.11.1	Detailed Description	191
6.11.2	Macro Definition Documentation	192
6.11.2.1	EXPECTED_FIFO_ELEMENTS	192
6.11.2.2	INPUT_FIFO_ELEMENTS	192
6.11.2.3	MODULUS_DIVISOR	192
6.11.2.4	OUTPUT_FIFO_ELEMENTS	193
6.11.2.5	SDRAM_FIFO_ELEMENTS	193
6.11.3	Function Documentation	193

6.11.3.1	<a href="#">get_fifo_status</a>	193
6.11.3.2	<a href="#">main</a>	193
6.11.4	<a href="#">Variable Documentation</a>	193
6.11.4.1	<a href="#">program_name</a>	194
6.12	<a href="#">examples/incenc_encoders.c File Reference</a>	194
6.12.1	<a href="#">Detailed Description</a>	194
6.12.2	<a href="#">Function Documentation</a>	195
6.12.2.1	<a href="#">main</a>	195
6.12.2.2	<a href="#">sigint_handler</a>	196
6.12.3	<a href="#">Variable Documentation</a>	196
6.12.3.1	<a href="#">exit_program</a>	196
6.12.3.2	<a href="#">program_name</a>	196
6.13	<a href="#">examples/incenc_interrupt.c File Reference</a>	196
6.13.1	<a href="#">Detailed Description</a>	197
6.13.2	<a href="#">Function Documentation</a>	198
6.13.2.1	<a href="#">main</a>	198
6.13.3	<a href="#">Variable Documentation</a>	198
6.13.3.1	<a href="#">program_name</a>	198
6.14	<a href="#">examples/incenc_status.c File Reference</a>	198
6.14.1	<a href="#">Detailed Description</a>	199
6.14.2	<a href="#">Function Documentation</a>	199
6.14.2.1	<a href="#">main</a>	199
6.14.3	<a href="#">Variable Documentation</a>	200
6.14.3.1	<a href="#">program_name</a>	200
6.15	<a href="#">examples/library_test.c File Reference</a>	200
6.15.1	<a href="#">Detailed Description</a>	201
6.15.2	<a href="#">Macro Definition Documentation</a>	202
6.15.2.1	<a href="#">DM7820_MAX_DMA_BUFFER_COUNT</a>	202
6.15.2.2	<a href="#">DM7820_MAX_DMA_BUFFER_SIZE</a>	202
6.15.3	<a href="#">Typedef Documentation</a>	202
6.15.3.1	<a href="#">initialization_state_t</a>	202
6.15.4	<a href="#">Enumeration Type Documentation</a>	202
6.15.4.1	<a href="#">initialization_state</a>	202
6.15.5	<a href="#">Function Documentation</a>	203
6.15.5.1	<a href="#">expect_failure_and_check</a>	203
6.15.5.2	<a href="#">expect_success</a>	203
6.15.5.3	<a href="#">main</a>	203
6.15.6	<a href="#">Variable Documentation</a>	204
6.15.6.1	<a href="#">bad_device_name</a>	204
6.15.6.2	<a href="#">boards</a>	205

6.15.6.3	device_count	205
6.15.6.4	init_state	205
6.15.6.5	program_name	205
6.16	examples/loopback.c File Reference	205
6.16.1	Detailed Description	206
6.16.2	Macro Definition Documentation	207
6.16.2.1	BUF_NUM	207
6.16.2.2	BUF_SIZE	207
6.16.2.3	DMA_BUF_SIZE	207
6.16.2.4	SAMPLES	207
6.16.3	Function Documentation	207
6.16.3.1	ISR	207
6.16.3.2	main	208
6.16.4	Variable Documentation	208
6.16.4.1	board	208
6.16.4.2	dma_done_interrupts	208
6.16.4.3	fifo0_empty_interrupts	208
6.16.4.4	program_name	209
6.17	examples/prgclk_clocks.c File Reference	209
6.17.1	Detailed Description	209
6.17.2	Function Documentation	210
6.17.2.1	main	210
6.17.3	Variable Documentation	210
6.17.3.1	program_name	210
6.18	examples/pwm_interrupt.c File Reference	210
6.18.1	Detailed Description	211
6.18.2	Function Documentation	212
6.18.2.1	main	212
6.18.3	Variable Documentation	212
6.18.3.1	program_name	212
6.19	examples/pwm_measure.c File Reference	212
6.19.1	Detailed Description	214
6.19.2	Macro Definition Documentation	214
6.19.2.1	BUF_NUM	214
6.19.2.2	BUF_SIZE	214
6.19.2.3	DMA_BUF_SIZE	215
6.19.2.4	SAMPLES	215
6.19.2.5	UTC_RATE	215
6.19.3	Function Documentation	215
6.19.3.1	ISR	215

6.19.3.2	main	215
6.19.3.3	sigint_handler	216
6.19.4	Variable Documentation	216
6.19.4.1	board	216
6.19.4.2	dma_done_interrupts	216
6.19.4.3	exit_program	216
6.19.4.4	program_name	216
6.20	examples/pwm_modulators.c File Reference	216
6.20.1	Detailed Description	217
6.20.2	Function Documentation	218
6.20.2.1	main	218
6.20.3	Variable Documentation	218
6.20.3.1	program_name	218
6.21	examples/stdio_digital_io.c File Reference	218
6.21.1	Detailed Description	219
6.21.2	Function Documentation	219
6.21.2.1	main	219
6.21.3	Variable Documentation	219
6.21.3.1	program_name	219
6.22	examples/stdio_peripheral_output.c File Reference	220
6.22.1	Detailed Description	220
6.22.2	Function Documentation	220
6.22.2.1	main	220
6.22.3	Variable Documentation	221
6.22.3.1	program_name	221
6.23	examples/stdio_strobe_signal.c File Reference	221
6.23.1	Detailed Description	222
6.23.2	Function Documentation	222
6.23.2.1	main	222
6.23.3	Variable Documentation	222
6.23.3.1	program_name	222
6.24	examples/tmrctr_interrupt.c File Reference	222
6.24.1	Detailed Description	223
6.24.2	Function Documentation	224
6.24.2.1	ISR	224
6.24.2.2	main	224
6.24.2.3	sigint_handler	224
6.24.3	Variable Documentation	225
6.24.3.1	exit_program	225
6.24.3.2	interrupts	225

6.24.3.3	program_name	225
6.25	examples/tmrctr_status.c File Reference	225
6.25.1	Detailed Description	226
6.25.2	Macro Definition Documentation	226
6.25.2.1	TIMER_A0_DIVISOR	226
6.25.2.2	TIMER_A1_DIVISOR	226
6.25.2.3	TIMER_A1_FREQUENCY	227
6.25.2.4	TIMER_A2_DIVISOR	227
6.25.3	Function Documentation	227
6.25.3.1	disable_timers	227
6.25.3.2	main	227
6.25.4	Variable Documentation	228
6.25.4.1	program_name	228
6.26	examples/tmrctr_timers.c File Reference	228
6.26.1	Detailed Description	228
6.26.2	Function Documentation	229
6.26.2.1	main	229
6.26.2.2	sigint_handler	229
6.26.3	Variable Documentation	230
6.26.3.1	exit_program	230
6.26.3.2	program_name	230
6.27	include/dm7820_driver.h File Reference	230
6.27.1	Detailed Description	233
6.28	include/dm7820_globals.h File Reference	234
6.28.1	Detailed Description	234
6.29	include/dm7820_ioctl.h File Reference	234
6.29.1	Detailed Description	236
6.30	include/dm7820_library.h File Reference	236
6.30.1	Detailed Description	241
6.31	include/dm7820_registers.h File Reference	241
6.31.1	Detailed Description	249
6.32	include/dm7820_types.h File Reference	249
6.32.1	Detailed Description	257



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

DM7820 driver header file . . . . .	7
DM7820 driver enumerations . . . . .	8
DM7820 driver macros . . . . .	9
DM7820 driver constants . . . . .	10
DM7820 driver structures . . . . .	11
DM7820 driver forward declarations . . . . .	12
DM7820 driver functions . . . . .	13
DM7820 global variable header file . . . . .	30
DM7820 ioctl header file . . . . .	31
DM7820 ioctl enumerations . . . . .	32
DM7820 ioctl structures . . . . .	33
DM7820 ioctl macros . . . . .	34
DM7820 user library header file . . . . .	35
DM7820 user library macros . . . . .	36
DM7820 user library type definitions . . . . .	39
DM7820 user library structures . . . . .	40
DM7820 user library functions . . . . .	41
DM7820 user library advanced interrupt functions . . . . .	42
DM7820 user library FIFO functions . . . . .	46
DM7820 user library general functions . . . . .	55
DM7820 user library incremental encoder functions . . . . .	60
DM7820 user library pulse width modulator functions . . . . .	66
DM7820 user library programmable clock functions . . . . .	70
DM7820 user library standard I/O functions . . . . .	73
DM7820 user library 8254 timer/counter functions . . . . .	77
DM7820 register header file . . . . .	80
DM7820 register BAR0 (memory-mapped PLX registers) offsets . . . . .	81
DM7820 register BAR1 (I/O-mapped PLX registers) offsets . . . . .	82
DM7820 register BAR2 (memory-mapped FPGA registers) offsets . . . . .	83
DM7820 register board control offsets . . . . .	87
DM7820 register standard I/O offsets . . . . .	88
DM7820 register 8254 timer/counter offsets . . . . .	89
DM7820 register FIFO 0 offsets . . . . .	90
DM7820 register FIFO 1 offsets . . . . .	91
DM7820 register programmable clock 0 offsets . . . . .	92
DM7820 register programmable clock 1 offsets . . . . .	93
DM7820 register programmable clock 2 offsets . . . . .	94

DM7820 register programmable clock 3 offsets . . . . .	95
DM7820 register advanced interrupt 0 offsets . . . . .	96
DM7820 register advanced interrupt 1 offsets . . . . .	97
DM7820 register incremental encoder 0 offsets . . . . .	98
DM7820 register incremental encoder 1 offsets . . . . .	99
DM7820 register pulse width modulator 0 offsets . . . . .	100
DM7820 register pulse width modulator 1 offsets . . . . .	101
DM7820 register 8254 timer/counter A offsets . . . . .	102
DM7820 register 8254 timer/counter B offsets . . . . .	103
DM7820 register PCI region lengths . . . . .	104
DM7820 register functional block identifiers . . . . .	105
DM7820 register functional block identifier values . . . . .	106
DM7820 register functional block identifier offsets . . . . .	107
DM7820 register BAR2 macros . . . . .	108
DM7820 register BAR2 FPGA Version Register . . . . .	109
DM7820 register BAR2 Board Reset Register . . . . .	110
DM7820 type definition header file . . . . .	111
DM7820 type enumerations . . . . .	112
DM7820 type PCI enumerations . . . . .	113
DM7820 type standard I/O enumerations . . . . .	114
DM7820 type timer/counter enumerations . . . . .	116
DM7820 type programmable clock enumerations . . . . .	120
DM7820 type pulse width modulator enumerations . . . . .	125
DM7820 type incremental encoder enumerations . . . . .	128
DM7820 type FIFO enumerations . . . . .	132
DM7820 type advanced interrupt enumerations . . . . .	137
DM7820 type interrupt enumerations . . . . .	139
DM7820 type definition structures . . . . .	142
DM7820 type definition typedefs . . . . .	143



## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_dm7820_interrupt_info</a>	Interrupt source information . . . . .	145
<a href="#">DM7820_Board_Descriptor</a>	DM7820 board descriptor. This structure holds information about a device needed by the library	146
<a href="#">dm7820_device_descriptor</a>	DM7820 device descriptor. This structure holds information about a device needed by the kernel	147
<a href="#">dm7820_dma_descriptor_t</a>	DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer . . . . .	149
<a href="#">dm7820_dma_function_arguments</a>	Structure encapsulating arguments to all possible DMA functions . . . . .	150
<a href="#">dm7820_dma_initialize_arguments</a>	Arguments for DMA initialization function . . . . .	151
<a href="#">dm7820_dma_list_item_t</a>	DM7820 DMA buffer list item . . . . .	151
<a href="#">dm7820_interrupt_control</a>	Structure containing information needed to acknowledge, disable, and enable a particular interrupt source . . . . .	152
<a href="#">dm7820_interrupt_status_source</a>	Interrupt source information for a single Interrupt Status Register bit . . . . .	153
<a href="#">dm7820_ioctl_argument</a>	ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call . . . . .	154
<a href="#">dm7820_ioctl_dma_function</a>	ioctl() request structure for performing a DMA function . . . . .	155
<a href="#">dm7820_ioctl_interrupt_status</a>	ioctl() request structure for getting interrupt status and waiting for an interrupt to occur . . . . .	156
<a href="#">dm7820_ioctl_region_modify</a>	ioctl() request structure for PCI region read/modify/write . . . . .	157
<a href="#">dm7820_ioctl_region_readwrite</a>	ioctl() request structure for read from or write to PCI region . . . . .	158
<a href="#">dm7820_minior_int_reg_layout</a>	Minor interrupt register bit layout . . . . .	159
<a href="#">dm7820_pci_access_request</a>	PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions . . . . .	160

[dm7820\\_pci\\_region](#)

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

161

[register\\_read](#) . . . . .

162

## Chapter 3

# File Index

### 3.1 File List

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

examples/ <a href="#">advint_event_interrupt.c</a>	Example program which demonstrates how to use advanced interrupt block event mode interrupts . . . . .	165
examples/ <a href="#">advint_match_interrupt.c</a>	Example program which demonstrates how to use advanced interrupt block match mode interrupts . . . . .	167
examples/ <a href="#">advint_status.c</a>	Example program which demonstrates how to get advanced interrupt status when not using interrupts . . . . .	169
examples/ <a href="#">advint_strobe_interrupt.c</a>	Example program which demonstrates how to use advanced interrupt block strobe mode interrupts . . . . .	171
examples/ <a href="#">basic_test.c</a>	Program which tests the basic functionality of the driver . . . . .	173
examples/ <a href="#">brdctl_device_info.c</a>	Example program which demonstrates how to obtain version information and PCI master capable status from a device . . . . .	179
examples/ <a href="#">dma_capture_buffers.c</a>	Example program which demonstrates how to use DMA to continuously acquire samples . . .	180
examples/ <a href="#">dma_capture_file.c</a>	Example program which demonstrates how to use DMA to continuously acquire samples . . .	183
examples/ <a href="#">fifo_cascaded.c</a>	Example program which demonstrates how to use the FIFO block FIFOs such that the output of one FIFO serves as the input of another FIFO . . . . .	186
examples/ <a href="#">fifo_interrupt.c</a>	Example program which demonstrates a loopback accross the FIFO's using DMA . . . . .	189
examples/ <a href="#">fifo_pci_access.c</a>	Example program which demonstrates how to use the FIFO block FIFOs with PCI read requests clocking data out of the FIFO and PCI write requests clocking data into the FIFO . . . . .	191
examples/ <a href="#">incenc_encoders.c</a>	Example program which demonstrates how to use the incremental encoder block encoders . .	194
examples/ <a href="#">incenc_interrupt.c</a>	Example program which demonstrates how to use the incremental encoder block interrupts . .	196
examples/ <a href="#">incenc_status.c</a>	Example program which demonstrates how to get incremental encoder status when not using interrupts . . . . .	198
examples/ <a href="#">library_test.c</a>	Program which tests the basic functionality of the user library . . . . .	200

examples/ <a href="#">loopback.c</a>	Example program which demonstrates how to use the FIFO block interrupts. Note: This example requires a loopback cable! . . . . .	205
examples/ <a href="#">prgclk_clocks.c</a>	Example program which demonstrates how to use the programmable clock block clocks . . . .	209
examples/ <a href="#">pwm_interrupt.c</a>	Example program which demonstrates how to use the pulse width modulator block interrupts .	210
examples/ <a href="#">pwm_measure.c</a>	This program demonstrates how an incoming pulse width can be measured using the DM7820	212
examples/ <a href="#">pwm_modulators.c</a>	Example program which demonstrates how to use the pulse width modulator block modulators	216
examples/ <a href="#">stdio_digital_io.c</a>	Example program which demonstrates how to use the standard I/O block digital I/O . . . . .	218
examples/ <a href="#">stdio_peripheral_output.c</a>	Example program which demonstrates how to select peripheral outputs for standard I/O block digital I/O ports . . . . .	220
examples/ <a href="#">stdio_strobe_signal.c</a>	Example program which demonstrates how to use the standard I/O block strobe signals . . . .	221
examples/ <a href="#">tmrctr_interrupt.c</a>	Example program which demonstrates how to use 8254 timer/counter block timer interrupts . .	222
examples/ <a href="#">tmrctr_status.c</a>	Example program which demonstrates how to get 8254 timer/counter block timer status when not using interrupts . . . . .	225
examples/ <a href="#">tmrctr_timers.c</a>	Example program which demonstrates how to use the 8254 timer/counter block timers . . . . .	228
include/ <a href="#">dm7820_driver.h</a>	Definitions for the DM7820 driver . . . . .	230
include/ <a href="#">dm7820_globals.h</a>	Global variables used both in the kernel and in the library . . . . .	234
include/ <a href="#">dm7820_ioctl.h</a>	Low level ioctl() request descriptor structure and request code definitions . . . . .	234
include/ <a href="#">dm7820_library.h</a>	DM7820 user library definitions . . . . .	236
include/ <a href="#">dm7820_registers.h</a>	Register definitions for DM7820 devices . . . . .	241
include/ <a href="#">dm7820_types.h</a>	Type definitions used both in kernel and user space . . . . .	249
include/ <a href="#">dm7820_version.h</a>		??

## Chapter 4

# Module Documentation

### 4.1 DM7820 driver header file

#### Modules

- [DM7820 driver enumerations](#)
- [DM7820 driver macros](#)
- [DM7820 driver constants](#)
- [DM7820 driver structures](#)
- [DM7820 driver forward declarations](#)
- [DM7820 driver functions](#)

#### 4.1.1 Detailed Description

## 4.2 DM7820 driver enumerations

### Typedefs

- typedef enum  
[dm7820\\_pci\\_region\\_access\\_dir](#) [dm7820\\_pci\\_region\\_access\\_dir\\_t](#)  
*Standard PCI region access direction type.*

### Enumerations

- enum [dm7820\\_pci\\_region\\_access\\_dir](#) { [DM7820\\_PCI\\_REGION\\_ACCESS\\_READ](#) = 0, [DM7820\\_PCI\\_REGION\\_ACCESS\\_WRITE](#) }  
*Direction of access to standard PCI region.*

#### 4.2.1 Detailed Description

#### 4.2.2 Enumeration Type Documentation

##### 4.2.2.1 enum [dm7820\\_pci\\_region\\_access\\_dir](#)

Direction of access to standard PCI region.

Enumerator:

- [DM7820\\_PCI\\_REGION\\_ACCESS\\_READ](#)** Read from the region  
**[DM7820\\_PCI\\_REGION\\_ACCESS\\_WRITE](#)** Write to the region

Definition at line 60 of file [dm7820\\_driver.h](#).

## 4.3 DM7820 driver macros

### Macros

- `#define DM7820_DEVICE_NAME_LENGTH 22`  
*Maximum number of characters in device's name.*
- `#define DM7820_PCI_DEVICE_ID 0x7820`  
*DM7820 PCI device ID.*
- `#define RTD_PCI_VENDOR_ID 0x1435`  
*RTD Embedded Technologies PCI vendor ID.*
- `#define DM7820_PCI_REGIONS PCI_ROM_RESOURCE`  
*Number of standard PCI regions.*
- `#define DM7820_FIFO_CHANNELS 2`  
*Number of FIFO channels per device.*
- `#define DM7820_MAX_DMA_BUFFER_SIZE 0x40000`  
*Maximum size in bytes of any DMA buffer.*
- `#define DM7820_MAX_DMA_BUFFER_COUNT 16`  
*Maximum number of DMA buffers per DMA/FIFO channel.*
- `#define DM7820_INT_QUEUE_SIZE 0x10`  
*Maximum number of entries in the interrupt status queue;.*

### 4.3.1 Detailed Description

DM7820\_Driver\_Enumerations

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 `#define DM7820_MAX_DMA_BUFFER_COUNT 16`

Maximum number of DMA buffers per DMA/FIFO channel.

#### Note

Be aware that the probability of DMA buffer allocation failure increases as the number of buffers per DMA/FIFO channel increases.

If this default value does not suit your needs, you can change it and then recompile the driver.

Definition at line 159 of file dm7820\_driver.h.

#### 4.3.2.2 `#define DM7820_MAX_DMA_BUFFER_SIZE 0x40000`

Maximum size in bytes of any DMA buffer.

#### Note

Be aware that the probability of DMA buffer allocation failure increases as the buffer size increases.

If this default value does not suit your needs, you can change it and then recompile the driver.

Definition at line 144 of file dm7820\_driver.h.

## 4.4 DM7820 driver constants

### 4.4.1 Detailed Description

DM7820\_Driver\_Macros



## 4.5 DM7820 driver structures

### Data Structures

- struct [dm7820\\_pci\\_region](#)  
*DM7820 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.*
- struct [dm7820\\_dma\\_descriptor\\_t](#)  
*DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.*
- struct [dm7820\\_dma\\_list\\_item\\_t](#)  
*DM7820 DMA buffer list item.*
- struct [dm7820\\_device\\_descriptor](#)  
*DM7820 device descriptor. This structure holds information about a device needed by the kernel.*
- struct [dm7820\\_interrupt\\_status\\_source](#)  
*Interrupt source information for a single Interrupt Status Register bit.*

### Typedefs

- typedef struct [dm7820\\_pci\\_region](#) [dm7820\\_pci\\_region\\_t](#)  
*DM7820 PCI region descriptor type.*
- typedef struct  
[dm7820\\_device\\_descriptor](#) [dm7820\\_device\\_descriptor\\_t](#)  
*DM7820 device descriptor type.*
- typedef struct  
[dm7820\\_interrupt\\_status\\_source](#) [dm7820\\_interrupt\\_status\\_source\\_t](#)  
*Interrupt Status Register bit interrupt source information type.*

#### 4.5.1 Detailed Description

DM7820\_Driver\_Constants

## 4.6 DM7820 driver forward declarations

### Variables

- static struct file\_operations [dm7820\\_file\\_ops](#)  
*File operations supported by driver.*

### 4.6.1 Detailed Description

DM7820\_Driver\_Structures

## 4.7 DM7820 driver functions

### Functions

- static void `dm7820_access_pci_region` (const `dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_pci_access_request_t` \*pci\_request, `dm7820_pci_region_access_dir_t` direction)  
*Read from or write to one of the standard PCI regions.*
- static int `dm7820_allocate_irq` (`dm7820_device_descriptor_t` \*dm7820\_device, const struct pci\_dev \*pci\_device)  
*Allocate an interrupt line for a DM7820 device.*
- static void `dm7820_disable_all_interrupts` (const `dm7820_device_descriptor_t` \*dm7820\_device)  
*Disable all non-PLX interrupts for the specified DM7820 device.*
- static void `dm7820_enable_plx_interrupts` (const `dm7820_device_descriptor_t` \*dm7820\_device, uint8\_t enable)  
*Disable or enable PLX interrupts for the specified DM7820 device.*
- static void `dm7820_free_dma_mappings` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)  
*Free all coherent/consistent DMA mappings for the given DMA/FIFO channel on the specified DM7820 device.*
- static int `dm7820_get_interrupt_status` (`dm7820_device_descriptor_t` \*dm7820\_device, unsigned long ioctl\_param)  
*Get interrupt status for the specified DM7820 device, optionally waiting for an interrupt to occur before returning the status.*
- static void `dm7820_get_pci_master_status` (`dm7820_device_descriptor_t` \*dm7820\_device, uint8\_t \*pci\_master)  
*Determine whether or not a device is PCI master capable.*
- static void `dm7820_initialize_device_descriptor` (`dm7820_device_descriptor_t` \*dm7820\_device)  
*Initialize the device descriptor for the specified DM7820 device.*
- static int `dm7820_initialize_dma` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_ioctl_argument_t` \*ioctl\_argument)  
*Initialize DMA for the specified DM7820 device.*
- `dma_addr_t` `dm7820_get_buffer_phy_addr` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)  
*Returns the physical address of the next available DMA buffer.*
- static int `dm7820_dma_read` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_ioctl_argument_t` \*ioctl\_argument)  
*Read from DMA buffer to copy to user.*
- static int `dm7820_dma_write` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_ioctl_argument_t` \*ioctl\_argument)  
*Write to DMA buffer.*
- static int `dm7820_dma_stop` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)  
*Stops a DMA transfer on the specified channel if one is currently running.*
- static int `dm7820_dma_pause` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)  
*Pause DMA – Used by the STOP\_DMA function.*
- static int `dm7820_dma_check_xfer` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)  
*Checks if there is a transfer currently underway for the specified DMA/FIFO channel.*
- static void `dm7820_initialize_hardware` (const `dm7820_device_descriptor_t` \*dm7820\_device)  
*Initialize the specified DM7820 device.*
- static void `dm7820_int_queue_add` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_interrupt_source` source)  
*Add an interrupt source to the queue.*
- static `dm7820_interrupt_info` `dm7820_dequeue_interrupt` (`dm7820_device_descriptor_t` \*dm7820\_device)

- Remove an interrupt from the front of the queue.*

  - static irqreturn\_t [dm7820\\_interrupt\\_handler](#) (int irq\_number, void \*device\_id)

*DM7820 device interrupt handler.*
- static long [dm7820\\_ioctl](#) (struct file \*file, unsigned int request\_code, unsigned long ioctl\_param)

*Process ioctl(2) system calls directed toward a DM7820 device file.*
- int [dm7820\\_load](#) (void)

*Perform all actions necessary to initialize the DM7820 driver and devices.*
- static int [dm7820\\_modify\\_pci\\_region](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)

*Read an unsigned value from one of a device's PCI regions, modify certain bits in the value, and then write it back to the region.*
- static int [dm7820\\_open](#) (struct inode \*inode, struct file \*file)

*Prepare a DM7820 device file to be opened and used.*
- static unsigned int [dm7820\\_poll](#) (struct file \*file, struct poll\_table\_struct \*poll\_table)

*Determine whether or not a DM7820 device is readable. This function supports the poll(2) and select(2) system calls.*
- static int [dm7820\\_probe\\_device\\_blocks](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device)

*Probe and set up all functional blocks on a device.*
- static int [dm7820\\_probe\\_devices](#) (uint32\_t \*device\_count, [dm7820\\_device\\_descriptor\\_t](#) \*\*device\_descriptors)

*Probe and set up all DM7820 devices.*
- static int [dm7820\\_process\\_pci\\_regions](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, const struct pci\_dev \*pci\_device)

*For each of the standard PCI regions, get the region's base address and length from kernel PCI resource information set up at boot. Also, remap any memory-mapped region into the kernel's virtual address space.*
- static int [dm7820\\_read\\_pci\\_region](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)

*Read an unsigned value from one of a device's PCI regions.*
- static int [dm7820\\_register\\_char\\_device](#) (int \*major)

*Register the DM7820 character device and request dynamic allocation of a character device major number.*
- static int [dm7820\\_release](#) (struct inode \*inode, struct file \*file)

*Do all processing necessary after the last reference to a DM7820 device file is released elsewhere in the kernel.*
- static void [dm7820\\_release\\_resources](#) (void)

*Release any resources allocated by the driver.*
- static int [dm7820\\_service\\_dma\\_function](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)

*Process user space DMA function request.*
- void [dm7820\\_unload](#) (void)

*Perform all actions necessary to deinitialize the DM7820 driver and devices.*
- static int [dm7820\\_unregister\\_char\\_device](#) (void)

*Unregister the DM7820 character device and free the character device major number.*
- static int [dm7820\\_validate\\_device](#) (const [dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device)

*Given what is assumed to be the address of a DM7820 device descriptor, make sure it corresponds to a valid DM7820 device descriptor.*
- static int [dm7820\\_validate\\_pci\\_access](#) (const [dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, const [dm7820\\_pci\\_access\\_request\\_t](#) \*pci\_request)

*Validate a user-space access to one of a device's PCI regions.*
- static int [dm7820\\_write\\_pci\\_region](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)

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

#### 4.7.1 Detailed Description

##### DM7820\_Driver\_Forward\_Declarations

## 4.7.2 Function Documentation

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

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

### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>pci_request</i>	Address of access' PCI request descriptor.
<i>direction</i>	Direction of access to PCI region (read from or write to).

### Warning

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

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

Allocate an interrupt line for a DM7820 device.

### Parameters

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

### Return values

0	Success.
< 0	

### Failure.

The following values may be returned:

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

### Note

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

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

Remove an interrupt from the front of the queue.

## Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
----------------------	---

## Return values

<i>dm7820_interrupt_info</i>	Information about the interrupt and the queue
------------------------------	---

## Note

None

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

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

## Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
----------------------	---

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

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

## Parameters

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

## Return values

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

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

Pause DMA – Used by the STOP\_DMA function.

## Parameters

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

## Return values

0	Success.
---	----------

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

Read from DMA buffer to copy to user.

## Parameters

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

## Return values

0	Success.
---	----------

## Note

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

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

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

## Parameters

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

## Return values

0	Success.
---	----------

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

Write to DMA buffer.

## Parameters

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

## Return values

0	Success.
---	----------

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

Disable or enable PLX interrupts for the specified DM7820 device.

## Parameters

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

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

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

#### Parameters

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

#### Note

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

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

Returns the physical address of the next available DMA buffer.

#### Parameters

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

#### Return values

0	Success.
---	----------

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

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

#### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>ioctl_param</i>	Third parameter given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

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



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

Determine whether or not a device is PCI master capable.

#### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>pci_master</i>	Address where pci master capable flag should be stored. Zero will be stored if the device is not PCI master capable. A non-zero value will be stored here if the device is PCI master capable.

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

Initialize the device descriptor for the specified DM7820 device.

#### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
----------------------	---

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

Initialize DMA for the specified DM7820 device.

#### Parameters

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

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

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

**Note**

When initializing DMA, this function: 1) allocates coherent/consistent DMA mappings, 2) allocates memory to store DMA buffer allocation information, 3) allocates memory to link DMA buffers into device's DMA buffer list, 4) links all DMA buffers into the device's DMA buffer list, 5) allocates memory to link DMA buffers in device's free DMA buffer list, and 6) links all DMA buffers into the device's free DMA buffer list.

Since a single DMA buffer must exist in physically contiguous memory, the probability of DMA buffer allocation failure increases as both the number of buffers to allocate and the size of each buffer increase.

Factors beyond the number and size of DMA buffers affect the probability of DMA buffer allocation failure. These factors include the number of processes on the system, how much system memory is already in use, and the presence of processes (such as the X server) which use a lot of memory.

System memory can be a scarce resource. Every system entity needs some amount of memory. Memory is being allocated and released all the time.

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

Initialize the specified DM7820 device.

**Parameters**

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
----------------------	---

**Note**

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

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

Add an interrupt source to the queue.

**Parameters**

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>source</i>	The source of the interrupt

**Note**

None

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

DM7820 device interrupt handler.

**Parameters**

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

**Return values**

<i>IRQ_HANDLED</i>	Interrupt successfully processed; 2.6 kernel only.
<i>IRQ_NONE</i>	Interrupt could not be processed; 2.6 kernel only.

**Note**

This function does not return a value on 2.4 kernels.

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

Process `ioctl(2)` system calls directed toward a DM7820 device file.

**Parameters**

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

**Return values**

0	Success.
< 0	0

Failure.

The following values may be returned:

- `-EINVAL` `request_code` is not valid.

Please see the descriptions of [dm7820\\_validate\\_device\(\)](#), [dm7820\\_read\\_pci\\_region\(\)](#), [dm7820\\_write\\_pci\\_region\(\)](#), and [dm7820\\_modify\\_pci\\_region\(\)](#) for information on other possible values returned in this case.

4.7.2.21 `int dm7820_load ( void )`

Perform all actions necessary to initialize the DM7820 driver and devices.

**Return values**

0	Success.
< 0	0

Failure.

The following values may be returned:

- `-ENOMEM` /proc entry creation failed.

Please see the descriptions of [dm7820\\_probe\\_devices\(\)](#) and [dm7820\\_register\\_char\\_device\(\)](#) for information on other possible values returned in this case.

**Note**

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

**4.7.2.22** `static int dm7820_modify_pci_region ( dm7820_device_descriptor_t * dm7820_device, unsigned long ioctl_param ) [static]`

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

#### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>ioctl_param</i>	Third parameter given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

- `-EFAULT` *ioctl\_param* is not a valid user address.

Please see the description of [dm7820\\_validate\\_pci\\_access\(\)](#) for information on other possible values returned in this case.

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

Prepare a DM7820 device file to be opened and used.

#### Parameters

<i>inode</i>	Address of kernel's inode descriptor for the device file.
<i>file</i>	Address of kernel's file descriptor for the device file.

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

- `-EBUSY` The device file is already open.
- `-ENODEV` The device's inode does not refer to a valid DM7820 device; 2.4 kernel only.

#### Note

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

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

Determine whether or not a DM7820 device is readable. This function supports the poll(2) and select(2) system calls.

## Parameters

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

## Return values

<i>status</i>	mask
---------------	------

Bit mask describing the status of the device.

The following bits may be set in the mask:

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

## Note

A DM7820 device is readable if and only if an interrupt just occurred on the device and a process has not yet obtained the interrupt status from it.

This function is used in the process of waiting until an interrupt occurs on a device.

This function can be executed before an interrupt occurs, which happens if something sends a signal to the process.

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

Probe and set up all functional blocks on a device.

## Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
----------------------	---

## Return values

<i>0</i>	Success.
<i>&lt;</i>	0

Failure.

The following values may be returned:

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

## Note

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

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

Probe and set up all DM7820 devices.

#### Parameters

<i>device_count</i>	Address where DM7820 device count should be stored. The content of this this memory is undefined if the function fails.
<i>device_descriptors</i>	Address where address of device descriptor memory should be stored. The content of this memory is undefined if the function fails.

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

- `-ENAMETOOLONG` Device name creation failed.
- `-ENODEV` No DM7820 devices found.
- `-ENOMEM` Device descriptor memory allocation failed.

Please see the descriptions of [dm7820\\_process\\_pci\\_regions\(\)](#), [dm7820\\_allocate\\_irq\(\)](#) ... for information on other possible values returned in this case.

#### Note

If set up of any device fails, then all device set up fails.

This function allocates memory for the DM7820 device descriptors based upon the number of devices found.

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

**4.7.2.27** `static int dm7820_process_pci_regions ( dm7820_device_descriptor_t * dm7820_device, const struct pci_dev * pci_device ) [static]`

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

#### Parameters

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

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

- `-EBUSY` I/O port or I/O memory range allocation failed.

- `-EIO` A region's resource flags are not valid.
- `-ENOMEM` Remapping a memory-mapped region into the kernel's virtual address space failed.

**Note**

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

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

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

**Parameters**

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>ioctl_param</i>	Third parameter given on <code>ioctl()</code> call. This is the user space address of the structure used to pass in the arguments.

**Return values**

<code>0</code>	Success.
<code>&lt; 0</code>	

Failure.

The following values may be returned:

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

Please see the description of [dm7820\\_validate\\_pci\\_access\(\)](#) for information on other possible values returned in this case.

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

Register the DM7820 character device and request dynamic allocation of a character device major number.

**Parameters**

<i>major</i>	Address where character device major number should be stored.
--------------	---

**Return values**

<code>0</code>	Success.
<code>&lt; 0</code>	

Failure.

The following values may be returned:

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

- `-ENOMEM` Memory allocation failed; returned by `alloc_chrdev_region()`.

#### Note

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

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

Do all processing necessary after the last reference to a DM7820 device file is released elsewhere in the kernel.

##### Parameters

<i>inode</i>	Address of kernel's inode descriptor for the device file. Unused.
<i>file</i>	Address of kernel's file descriptor for the device file.

##### Return values

<code>0</code>	Success.
<code>&lt; 0</code>	

Failure. Please see the description of [dm7820\\_validate\\_device\(\)](#) for information on possible values returned in this case.

#### Note

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

#### 4.7.2.31 `static void dm7820_release_resources ( void ) [static]`

Release any resources allocated by the driver.

#### Note

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

#### 4.7.2.32 `static int dm7820_service_dma.function ( dm7820_device_descriptor_t * dm7820_device, unsigned long ioctl_param ) [static]`

Process user space DMA function request.

##### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>ioctl_param</i>	Third parameter given on <code>ioctl()</code> call. This is the user space address of the structure used to pass in the arguments.

##### Return values

<code>0</code>	Success.
<code>&lt; 0</code>	

Failure.



The following values may be returned:

- `-EFAULT` `ioctl_param` is not a valid user address.
- `-EINVAL` DMA/FIFO channel to operate upon is not valid.
- `-ENOSYS` DMA function request is not valid.

Please see the descriptions of [dm7820\\_initialize\\_dma\(\)](#), and `????????` for information on other possible values returned in this case.

#### 4.7.2.33 `static int dm7820_unregister_char_device ( void ) [static]`

Unregister the DM7820 character device and free the character device major number.

##### Return values

<code>0</code>	Success.
<code>&lt; 0</code>	

Failure.

The following values may be returned:

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

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

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

##### Note

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

This function does not fail on 2.6 kernels.

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

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

##### Parameters

<code>dm7820_device</code>	Address of device descriptor to be verified.
----------------------------	--

##### Return values

<code>0</code>	Success.
<code>&lt; 0</code>	

Failure.

The following values may be returned:

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

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

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

#### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>pci_request</i>	Address of PCI region access request descriptor.

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

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

#### Note

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

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

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

#### Parameters

<i>dm7820_device</i>	Address of device's DM7820 device descriptor.
<i>ioctl_param</i>	Third parameter given on ioctl() call. This is the user space address of the structure used to pass in the arguments.

#### Return values

0	Success.
< 0	

Failure.

The following values may be returned:

- `-EFAULT` *ioctl\_param* is not a valid user address.

Please see the description of [dm7820\\_validate\\_pci\\_access\(\)](#) for information on other possible values returned in this case.

## 4.8 DM7820 global variable header file

### Variables

- static [dm7820\\_interrupt\\_control\\_t](#) `dm7820_interrupt_control` []  
*Table of information needed to acknowledge, disable, and enable all interrupt sources.*
- [dm7820\\_minor\\_int\\_reg\\_layout\\_t](#) `dm7820_minor_int_reg_layout` []  
*Table of information providing layout of all minor interrupt registers.*

### 4.8.1 Detailed Description

## 4.9 DM7820 ioctl header file

### Modules

- [DM7820 ioctl enumerations](#)
- [DM7820 ioctl structures](#)
- [DM7820 ioctl macros](#)

### 4.9.1 Detailed Description

## 4.10 DM7820 ioctl enumerations

### Typedefs

- typedef enum  
[dm7820\\_dma\\_manage\\_function](#) [dm7820\\_dma\\_manage\\_function\\_t](#)  
*Functions supported by driver DMA management system.*

### Enumerations

- enum [dm7820\\_dma\\_manage\\_function](#) {  
[DM7820\\_DMA\\_FUNCTION\\_INITIALIZE](#) = 0, [DM7820\\_DMA\\_FUNCTION\\_STOP](#), [DM7820\\_DMA\\_FUNCTION\\_READ](#), [DM7820\\_DMA\\_FUNCTION\\_WRITE](#),  
[DM7820\\_DMA\\_GET\\_BUFFER\\_ADDR](#) }  
*Functions supported by driver DMA management system.*

#### 4.10.1 Detailed Description

#### 4.10.2 Enumeration Type Documentation

##### 4.10.2.1 enum dm7820\_dma\_manage\_function

Functions supported by driver DMA management system.

Enumerator:

***DM7820\_DMA\_FUNCTION\_INITIALIZE*** DMA initialization  
***DM7820\_DMA\_FUNCTION\_STOP*** DMA stop  
***DM7820\_DMA\_FUNCTION\_READ*** DMA read  
***DM7820\_DMA\_FUNCTION\_WRITE*** DMA write  
***DM7820\_DMA\_GET\_BUFFER\_ADDR*** Get Address of DMA Buffer

Definition at line 58 of file dm7820\_ioctl.h.

## 4.11 DM7820 ioctl structures

### Data Structures

- struct [dm7820\\_ioctl\\_region\\_readwrite](#)  
*ioctl() request structure for read from or write to PCI region*
- struct [dm7820\\_ioctl\\_region\\_modify](#)  
*ioctl() request structure for PCI region read/modify/write*
- struct [dm7820\\_ioctl\\_interrupt\\_status](#)  
*ioctl() request structure for getting interrupt status and waiting for an interrupt to occur*
- struct [dm7820\\_dma\\_initialize\\_arguments](#)  
*Arguments for DMA initialization function.*
- union [dm7820\\_dma\\_function\\_arguments](#)  
*Structure encapsulating arguments to all possible DMA functions.*
- struct [dm7820\\_ioctl\\_dma\\_function](#)  
*ioctl() request structure for performing a DMA function*
- union [dm7820\\_ioctl\\_argument](#)  
*ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.*

### Typedefs

- typedef struct  
[dm7820\\_ioctl\\_region\\_readwrite](#) dm7820\_ioctl\_region\_readwrite\_t
- typedef struct  
[dm7820\\_ioctl\\_region\\_modify](#) dm7820\_ioctl\_region\_modify\_t  
*ioctl() PCI region read/modify/write request descriptor type*
- typedef struct  
[dm7820\\_ioctl\\_interrupt\\_status](#) dm7820\_ioctl\_interrupt\_status\_t  
*ioctl() interrupt status request descriptor type*
- typedef struct  
[dm7820\\_dma\\_initialize\\_arguments](#) dm7820\_dma\_initialize\_arguments\_t  
*Arguments for DMA initialization function.*
- typedef union  
[dm7820\\_dma\\_function\\_arguments](#) dm7820\_dma\_function\_arguments\_t  
*Structure encapsulating arguments to all possible DMA functions.*
- typedef struct  
[dm7820\\_ioctl\\_dma\\_function](#) dm7820\_ioctl\_dma\_function\_t  
*ioctl() request structure for performing a DMA function*
- typedef union [dm7820\\_ioctl\\_argument](#) dm7820\_ioctl\_argument\_t  
*ioctl() request descriptor type*

#### 4.11.1 Detailed Description

DM7820\_ioctl\_Enumerations

#### 4.11.2 Typedef Documentation

##### 4.11.2.1 typedef struct dm7820\_ioctl\_region\_readwrite dm7820\_ioctl\_region\_readwrite\_t

typedef for the PCI region access request type

Definition at line 120 of file dm7820\_ioctl.h.

## 4.12 DM7820 ioctl macros

### Macros

- `#define DM7820_IOCTL_MAGIC 'D'`  
*Unique 8-bit value used to generate unique ioctl() request codes.*
- `#define DM7820_IOCTL_REQUEST_BASE 0x00`  
*First ioctl() request number.*
- `#define DM7820_IOCTL_REGION_READ`  
*ioctl() request code for reading from a PCI region*
- `#define DM7820_IOCTL_REGION_WRITE`  
*ioctl() request code for writing to a PCI region*
- `#define DM7820_IOCTL_REGION_MODIFY`  
*ioctl() request code for PCI region read/modify/write*
- `#define DM7820_IOCTL_GET_INTERRUPT_STATUS`  
*ioctl() request code for getting interrupt status and waiting for an interrupt to occur*
- `#define DM7820_IOCTL_DMA_FUNCTION`  
*ioctl() request code for DMA function*
- `#define DM7820_IOCTL_WAKEUP`  
*ioctl() request code for User ISR thread wake up*
- `#define DM7820_IOCTL_INTERRUPT_INFO`  
*ioctl() request code to retrieve interrupt status information*

### 4.12.1 Detailed Description

DM7820\_ioctl\_Structures



## 4.13 DM7820 user library header file

### Modules

- [DM7820 user library macros](#)
- [DM7820 user library type definitions](#)
- [DM7820 user library structures](#)
- [DM7820 user library functions](#)

### 4.13.1 Detailed Description

## 4.14 DM7820 user library macros

### Macros

- `#define DM7820_Return_Status(status, string) if(status != 0) { error(EXIT_FAILURE,errno, "ERROR: %s FAILED\n",string); }`
- `#define DM7820_DMA_DEMAND_OFF_PCI_TO_DM7820 0x00`
- `#define DM7820_DMA_DEMAND_OFF_DM7820_TO_PCI 0x01`
- `#define DM7820_DMA_DEMAND_ON_PCI_TO_DM7820 0x02`
- `#define DM7820_DMA_DEMAND_ON_DM7820_TO_PCI 0x03`
- `#define DM7820_INCENC_DISABLE_PHASE_FILTER_TRANSITION(filter, transition) ((filter) |= (1 << (transition)))`  
*Configure an incremental encoder phase filter so that counter value update is disabled for the given transition.*
- `#define DM7820_INCENC_ENABLE_PHASE_FILTER_TRANSITION(filter, transition) ((filter) &= ~(1 << (transition)))`  
*Configure an incremental encoder phase filter so that counter value update is enabled for the given transition.*
- `#define DM7820_INCENC_RESET_PHASE_FILTER(filter) ((filter) = 0x00)`  
*Reset an incremental encoder phase filter.*
- `#define DM7820_INTERRUPT_STATUS_IS_SOURCE_PENDING(status, source) (((status) & (0x1LL << (source))) ? 0xFF : 0x00)`  
*Determine whether or not the specified interrupt source is pending in the interrupt status obtained via [DM7820\\_General\\_Get\\_Interrupt\\_Status\(\)](#).*

### 4.14.1 Detailed Description

### 4.14.2 Macro Definition Documentation

#### 4.14.2.1 `#define DM7820_DMA_DEMAND_OFF_DM7820_TO_PCI 0x01`

Demand Mode Off, DM7820 to PCI

Definition at line 69 of file `dm7820_library.h`.

#### 4.14.2.2 `#define DM7820_DMA_DEMAND_OFF_PCI_TO_DM7820 0x00`

Demand Mode Off, PCI to DM7820

Definition at line 63 of file `dm7820_library.h`.

#### 4.14.2.3 `#define DM7820_DMA_DEMAND_ON_DM7820_TO_PCI 0x03`

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

Definition at line 81 of file `dm7820_library.h`.

Referenced by `do_dma()`, and `main()`.

#### 4.14.2.4 `#define DM7820_DMA_DEMAND_ON_PCI_TO_DM7820 0x02`

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

Definition at line 75 of file `dm7820_library.h`.

Referenced by `main()`.

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

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

#### Parameters

<i>filter</i>	The phase filter to disable transition counter update in.
<i>transition</i>	The transition to disable counter update for.

Definition at line 99 of file dm7820\_library.h.

Referenced by main().

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

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

#### Parameters

<i>filter</i>	The phase filter to enable transition counter update in.
<i>transition</i>	The transition to enable counter update for.

Definition at line 118 of file dm7820\_library.h.

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

Reset an incremental encoder phase filter.

#### Parameters

<i>filter</i>	The phase filter to reset.
---------------	----------------------------

#### Note

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

Definition at line 136 of file dm7820\_library.h.

Referenced by main().

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

Determine whether or not the specified interrupt source is pending in the interrupt status obtained via [DM7820\\_General\\_Get\\_Interrupt\\_Status\(\)](#).

#### Parameters

<i>status</i>	Interrupt status to examine.
<i>source</i>	Interrupt source to determine state of.

#### Return values

<i>0x00</i>	The specified interrupt source is not pending.
<i>0xFF</i>	The specified interrupt source is pending.

Definition at line 165 of file dm7820\_library.h.

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

Check library function return status

Definition at line 56 of file dm7820\_library.h.

Referenced by clean\_up(), disable\_timers(), do\_8254(), do\_digital\_io(), do\_dma(), do\_fifo(), do\_incenc(), do\_pwm(), ISR(), and main().

## 4.15 DM7820 user library type definitions

### Typedefs

- typedef int [DM7820\\_Error](#)  
*DM7820 user library error code type.*
- typedef uint8\_t [dm7820\\_incenc\\_phase\\_filter](#)  
*Incremental encoder phase filter type.*

### 4.15.1 Detailed Description

DM7820\_Library\_Macros

## 4.16 DM7820 user library structures

### Data Structures

- struct [DM7820\\_Board\\_Descriptor](#)

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

### Typedefs

- typedef struct  
[DM7820\\_Board\\_Descriptor](#) [DM7820\\_Board\\_Descriptor](#)

#### 4.16.1 Detailed Description

DM7820\_Library\_Types

#### 4.16.2 Typedef Documentation

##### 4.16.2.1 typedef struct DM7820\_Board\_Descriptor DM7820\_Board\_Descriptor

DM7820 board descriptor type

Definition at line 239 of file dm7820\_library.h.

## 4.17 DM7820 user library functions

### Modules

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

### 4.17.1 Detailed Description

DM7820\_Library\_Structures

## 4.18 DM7820 user library advanced interrupt functions

### Functions

- [DM7820\\_Error DM7820\\_AdvInt\\_Get\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [uint8\\_t](#) \*occurred)  
*Determine whether or not a status condition has occurred for the given advanced interrupt.*
- [DM7820\\_Error DM7820\\_AdvInt\\_Read\\_Capture](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) \*value)  
*Read the capture register value for the given advanced interrupt and standard I/O port.*
- [DM7820\\_Error DM7820\\_AdvInt\\_Set\\_Compare](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) value)  
*Load the compare register for the given advanced interrupt and standard I/O port.*
- [DM7820\\_Error DM7820\\_AdvInt\\_Set\\_Mask](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) value)  
*Load the mask register for the given advanced interrupt and standard I/O port.*
- [DM7820\\_Error DM7820\\_AdvInt\\_Set\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [dm7820\\_advint\\_master\\_clock](#) master)  
*Select the master clock for the given advanced interrupt.*
- [DM7820\\_Error DM7820\\_AdvInt\\_Set\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [dm7820\\_advint\\_mode](#) mode)  
*Set the mode for the given advanced interrupt.*

### 4.18.1 Detailed Description

### 4.18.2 Function Documentation

#### 4.18.2.1 [DM7820\\_Error DM7820\\_AdvInt\\_Get\\_Status](#) ( [DM7820\\_Board\\_Descriptor](#) \* handle, [dm7820\\_advint\\_interrupt](#) interrupt, [uint8\\_t](#) \* occurred )

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

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> interrupt is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

#### Warning

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



**Note**

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

Referenced by `main()`.

#### 4.18.2.2 **DM7820\_Error DM7820\_AdvInt\_Read\_Capture** ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_advint\_interrupt** *interrupt*, **DM7820\_StdIO\_Port** *port*, **uint16\_t** \* *value* )

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

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> interrupt is not valid.</li> <li>• <code>EINVAL</code> port is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

Referenced by `main()`.

#### 4.18.2.3 **DM7820\_Error DM7820\_AdvInt\_Set\_Compare** ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_advint\_interrupt** *interrupt*, **DM7820\_StdIO\_Port** *port*, **uint16\_t** *value* )

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

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> interrupt is not valid.</li> <li>• <code>EINVAL</code> port is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

Referenced by `main()`.

#### 4.18.2.4 **DM7820\_Error** DM7820\_AdvInt\_Set\_Mask ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_advint\_interrupt** *interrupt*, **DM7820\_StdIO\_Port** *port*, **uint16\_t** *value* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL interrupt is not valid.</li> <li>EINVAL port is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

##### Warning

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

Referenced by main().

#### 4.18.2.5 **DM7820\_Error** DM7820\_AdvInt\_Set\_Master ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_advint\_interrupt** *interrupt*, **dm7820\_advint\_master\_clock** *master* )

Select the master clock for the given advanced interrupt.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL interrupt is not valid.</li> <li>EINVAL master is not valid.</li> <li>EOPNOTSUPP master is equal to DM7820_ADVINT_MASTER_RESERVED.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

**Note**

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

**Warning**

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

Referenced by `main()`.

**4.18.2.6 DM7820\_Error DM7820\_AdvInt\_Set\_Mode ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_advint\_interrupt *interrupt*, dm7820\_advint\_mode *mode* )**

Set the mode for the given advanced interrupt.

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"><li>• <code>EINVAL</code> interrupt is not valid.</li><li>• <code>EINVAL</code> mode is not valid.</li></ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

Referenced by `main()`.

## 4.19 DM7820 user library FIFO functions

### Functions

- [DM7820\\_Error DM7820\\_FIFO\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint8\\_t](#) enable)  
*Enable or disable the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Get\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_status\\_condition](#) condition, [uint8\\_t](#) \*occurred)  
*Determine whether or not the specified status condition has occurred for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Initialize](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint32\\_t](#) buffer\_count, [uint32\\_t](#) buffer\_size)  
*Set up direct memory access (DMA) for the given DMA/FIFO channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Create\\_Buffer](#) ([uint16\\_t](#) \*\*buf, [uint32\\_t](#) size)  
*Creates a user space DMA buffer.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Free\\_Buffer](#) ([uint16\\_t](#) \*\*buf, [uint32\\_t](#) size)  
*Frees a previously created user space buffer.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Read](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [void](#) \*user\_buffer, [uint32\\_t](#) num\_bufs)  
*Reads the DMA buffers in the driver.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Write](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [void](#) \*user\_buffer, [uint32\\_t](#) num\_bufs)  
*Copies a user buffer to DMA buffers to be sent into a FIFO.*
- [DM7820\\_Error DM7820\\_Stop\\_DMA](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo)  
*Aborts a DMA transfer on a given channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Configure](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint8\\_t](#) direction, [uint32\\_t](#) transfer\_size)  
*Configure the specified DMA channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint8\\_t](#) enable, [uint8\\_t](#) start)  
*Enable and/or Start a DMA channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_Read](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint16\\_t](#) \*data)  
*Read a single value from the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_DMA\\_Request](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_dma\\_request](#) source)  
*Set DMA request source for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_Data\\_Input](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_data\\_input](#) input)  
*Set data input for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_Input\\_Clock](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_input\\_clock](#) clock)  
*Set input clock for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_Output\\_Clock](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_output\\_clock](#) clock)  
*Set output clock for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Write](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint16\\_t](#) data)  
*Write a single value to the given FIFO.*

### 4.19.1 Detailed Description

DM7820\_Library\_AdvInt\_Functions

## 4.19.2 Function Documentation

### 4.19.2.1 DM7820\_Error DM7820\_FIFO\_DMA.Configure ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_fifo\_queue *fifo*, uint8\_t *direction*, uint32\_t *transfer\_size* )

Configure the specified DMA channel.

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>fifo</i> is not valid.</li> <li>EINVAL <i>direction</i> is not valid.</li> <li>ENOTTY <i>ioctl</i> call is invalid.</li> <li>EFAULT could get <i>ioctl_arguments</i> from userspace.</li> </ul> Please see the <i>ioctl(2)</i> man page for information on other possible values <i>errno</i> may have in this case.

Referenced by *do\_dma()*, and *main()*.

### 4.19.2.2 DM7820\_Error DM7820\_FIFO\_DMA.Create\_Buffer ( uint16\_t \*\* *buf*, uint32\_t *size* )

Creates a user space DMA buffer.

#### Parameters

<i>buf</i>	Pointer to the buffer to be set by malloc.
<i>size</i>	The size of the buffer to allocate.

#### Return values

0	Success
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>ENOMEM Memory allocation for the buffer failed, or memory was not mapped correctly for lock/unlock, or an attempt was made to lock more memory than is allowed.</li> <li>ENOSYS The implementation does not support memory locking.</li> <li>EAGAIN Some or all of the memory could not be locked.</li> <li>EINVAL The memory size was not a multiple of (PAGESIZE).</li> <li>EPERM The calling process did not have appropriate privileges to perform memory locking.</li> </ul>

Referenced by *main()*.

#### 4.19.2.3 **DM7820\_Error** DM7820\_FIFO\_DMA.Enable ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_fifo\_queue** *fifo*, **uint8\_t** *enable*, **uint8\_t** *start* )

Enable and/or Start a DMA channel.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>fifo</i> is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by clean\_up(), and main().

#### 4.19.2.4 **DM7820\_Error** DM7820\_FIFO\_DMA.Free\_Buffer ( **uint16\_t** \*\* *buf*, **uint32\_t** *size* )

Frees a previously created user space buffer.

##### Parameters

<i>buf</i>	A pointer to the buffer to be released.
<i>size</i>	Size of the buffer to be released.

##### Return values

0	Success
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>ENOMEM Memory allocation for the buffer failed, or memory was not mapped correctly for lock/unlock.</li> <li>ENOSYS The implementation does not support memory locking/unlocking.</li> <li>EAGAIN Some or all of the memory could not be unlocked.</li> <li>EINVAL The memory size was not a multiple of (PAGESIZE).</li> <li>EPERM The calling process did not have appropriate privileges to perform memory unlocking.</li> </ul>

Referenced by main().

#### 4.19.2.5 **DM7820\_Error** DM7820\_FIFO\_DMA.Initialize ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_fifo\_queue** *fifo*, **uint32\_t** *buffer\_count*, **uint32\_t** *buffer\_size* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EAGAIN DMA has already been initialized for fifo.</li> <li>• EINVAL fifo is not valid.</li> <li>• EINVAL buffer_count is zero.</li> <li>• EINVAL buffer_count exceeds the default value DM7820_MAX_DMA_BUFFER_COUNT.</li> <li>• EINVAL buffer_size is zero.</li> <li>• EINVAL buffer_values exceeds the default value DM7820_MAX_DMA_BUFFER_SIZE.</li> <li>• ENOMEM Kernel memory allocation failed.</li> <li>• EOPNOTSUPP The device is not PCI master capable.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

## Note

Since a single DMA buffer must exist in physically contiguous memory, the probability of DMA buffer allocation failure increases as both the number of buffers to allocate and the size of each buffer increase.

Factors beyond the number and size of DMA buffers affect the probability of DMA buffer allocation failure. These factors include the number of processes on the system, how much system memory is already in use, and the presence of processes (such as the X server) which use a lot of memory.

System memory can be a scarce resource. Every system entity needs some amount of memory. Memory is being allocated and released all the time.

The default value for DM7820\_MAX\_DMA\_BUFFER\_COUNT is 16. If you need to change this, edit [include/dm7820\\_driver.h](#), save the changes, recompile the driver, and reload the driver.

The default value for DM7820\_MAX\_DMA\_BUFFER\_SIZE is 262,144 bytes (256 kilobytes). If you need to change this, edit [include/dm7820\\_driver.h](#), save the changes, recompile the driver, and reload the driver.

As the application designer, you have some flexibility to configure DMA as your purpose suits. However, if this function fails with errno ENOMEM, you need to decrease the number of buffers requested or ask for smaller buffers. In extreme cases, you may need to do both.

The values you specify here are used for all DMA transfers afterward.

Referenced by `do_dma()`, and `main()`.

#### 4.19.2.6 DM7820\_Error DM7820\_FIFO\_DMA\_Read ( DM7820\_Board\_Descriptor \* handle, dm7820\_fifo\_queue fifo, void \* user\_buffer, uint32\_t num\_bufs )

Reads the DMA buffers in the driver.

## Parameters

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

## Return values

0	Success
-1	Failure

Referenced by main().

**4.19.2.7 DM7820\_Error DM7820\_FIFO\_DMA\_Write** ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_fifo\_queue** *fifo*, **void** \* *user\_buffer*, **uint32\_t** *num\_bufs* )

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

## Parameters

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

## Return values

0	Success
-1	Failure

Referenced by main().

**4.19.2.8 DM7820\_Error DM7820\_FIFO\_Enable** ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_fifo\_queue** *fifo*, **uint8\_t** *enable* )

Enable or disable the given FIFO.

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>fifo</i> is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.



**Note**

The hardware clears a FIFO when it is disabled.

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

#### 4.19.2.9 DM7820\_Error DM7820\_FIFO\_Get\_Status ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_fifo\_queue *fifo*, dm7820\_fifo\_status\_condition *condition*, uint8\_t \* *occurred* )

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

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>fifo</i> is not valid.</li> <li>EINVAL <i>condition</i> is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

**Warning**

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

**Note**

If *condition* is `DM7820_FIFO_STATUS_FULL` or `DM7820_FIFO_STATUS_EMPTY`, this function clears the board's corresponding FIFO status flag before reading the FIFO status to get accurate status. If the condition is still pending after the clear, the hardware will reassert the status flag.

If *condition* is `DM7820_FIFO_STATUS_READ_REQUEST`, `DM7820_FIFO_STATUS_WRITE_REQUEST`, `DM7820_FIFO_STATUS_OVERFLOW`, or `DM7820_FIFO_STATUS_UNDERFLOW`, this function reads the FIFO status and then clears the board's corresponding FIFO status flag if it is set. The hardware will not reassert the flag until the next time the condition occurs.

Referenced by `get_fifo_status()`, and `main()`.

#### 4.19.2.10 DM7820\_Error DM7820\_FIFO\_Read ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_fifo\_queue *fifo*, uint16\_t \* *data* )

Read a single value from the given FIFO.

**Parameters**

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> fifo is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `main()`.

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

Set data input for the given FIFO.

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> fifo is not valid.</li> <li>• <code>EINVAL</code> input is not valid.</li> <li>• <code>EOPNOTSUPP</code> input is not supported by fifo.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

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

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

Set DMA request source for the given FIFO.

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> fifo is not valid.</li> <li>• <code>EINVAL</code> source is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

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

#### 4.19.2.13 DM7820\_Error DM7820\_FIFO\_Set\_Input\_Clock ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_fifo\_queue *fifo*, dm7820\_fifo\_input\_clock *clock* )

Set input clock for the given FIFO.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL <i>fifo</i> is not valid.</li> <li>• EINVAL <i>clock</i> is not valid.</li> <li>• EOPNOTSUPP <i>clock</i> is equal to DM7820_FIFO_INPUT_CLOCK_RESERVED_1, DM7820_FIFO_INPUT_CLOCK_RESERVED_2, DM7820_FIFO_INPUT_CLOCK_RESERVED_3, or DM7820_FIFO_INPUT_CLOCK_RESERVED_4.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

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

#### 4.19.2.14 DM7820\_Error DM7820\_FIFO\_Set\_Output\_Clock ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_fifo\_queue *fifo*, dm7820\_fifo\_output\_clock *clock* )

Set output clock for the given FIFO.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL <i>fifo</i> is not valid.</li> <li>• EINVAL <i>clock</i> is not valid.</li> <li>• EOPNOTSUPP <i>clock</i> is equal to DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3, or DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

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

#### 4.19.2.15 **DM7820\_Error** DM7820\_FIFO\_Write ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_fifo\_queue** *fifo*, **uint16\_t** *data* )

Write a single value to the given FIFO.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>fifo</i> is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

#### 4.19.2.16 **DM7820\_Error** DM7820\_Stop\_DMA ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_fifo\_queue** *fifo* )

Aborts a DMA transfer on a given channel.

##### Parameters

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

##### Return values

0	Success
-1	Failure

## 4.20 DM7820 user library general functions

### Functions

- [DM7820\\_Error DM7820\\_General\\_Close\\_Board](#) ([DM7820\\_Board\\_Descriptor](#) \*handle)  
*Close a DM7820 device file.*
- [DM7820\\_Error DM7820\\_General\\_Enable\\_Interrupt](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_interrupt\\_source](#) source, [uint8\\_t](#) enable)  
*Enable or disable the given interrupt source.*
- [DM7820\\_Error DM7820\\_General\\_Get\\_Interrupt\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_interrupt\\_info](#) \*interrupt\_info, [uint8\\_t](#) wait\_for\_interrupt)  
*Get a device's interrupt status, optionally waiting for an interrupt to occur.*
- [DM7820\\_Error DM7820\\_General\\_Open\\_Board](#) ([uint8\\_t](#) dev\_num, [DM7820\\_Board\\_Descriptor](#) \*\*handle)  
*Open a DM7820 device file.*
- [DM7820\\_Error DM7820\\_General\\_Get\\_Version\\_Info](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [uint8\\_t](#) \*fpga\_type\_id, [uint8\\_t](#) \*fpga\_version, [uint16\\_t](#) \*svn\_version)  
*Read a device's FPGA and source code revision control versions.*
- [DM7820\\_Error DM7820\\_General\\_Is\\_PCI\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [uint8\\_t](#) \*pci\_master)  
*Determine whether or not a device is PCI master capable.*
- [DM7820\\_Error DM7820\\_General\\_Reset](#) ([DM7820\\_Board\\_Descriptor](#) \*handle)  
*Reset a DM7820 device.*
- [DM7820\\_Error DM7820\\_General\\_RemoveISR](#) ([DM7820\\_Board\\_Descriptor](#) \*handle)  
*Uninstall userspace ISR.*
- [DM7820\\_Error DM7820\\_General\\_StartThread](#) ([int](#)(\*fnct)([void](#) \*), [void](#) \*data)  
*Creates thread to watch for interrupts and call userspace ISR.*
- [void](#) \* [DM7820\\_General\\_WaitForInterrupt](#) ([void](#) \*ptr)  
*Waits for DMA Done interrupts.*
- [DM7820\\_Error DM7820\\_General\\_SetISRPriority](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [int](#) priority)  
*Changes the Priority for the ISR thread.*

### 4.20.1 Detailed Description

DM7820\_Library\_FIFO\_Functions

### 4.20.2 Function Documentation

#### 4.20.2.1 [DM7820\\_Error DM7820\\_General\\_Close\\_Board](#) ( [DM7820\\_Board\\_Descriptor](#) \* handle )

Close a DM7820 device file.

#### Parameters

<i>handle</i>	Address of device's library board descriptor.
---------------	---

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• ENODATA handle is NULL.</li> </ul> Please see the close(2) man page for information on other possible values errno may have in this case.

**Note**

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

**Warning**

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

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

#### 4.20.2.2 **DM7820\_Error DM7820\_General\_Enable\_Interrupt** ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_interrupt\_source** *source*, **uint8\_t** *enable* )

Enable or disable the given interrupt source.

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL source is not valid.</li> <li>EOPNOTSUPP source is valid but cannot be modified from user space.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

**Note**

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

Referenced by `main()`.

#### 4.20.2.3 **DM7820\_Error DM7820\_General\_Get\_Interrupt\_Status** ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_interrupt\_info** \* *interrupt\_info*, **uint8\_t** *wait\_for\_interrupt* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINTR</code> The process was waiting for an interrupt but received a signal before an interrupt occurred. This is not a fatal error but rather means the wait should be retried.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

## Note

If this function is being used to wait for interrupts, signals can wake up the process before an interrupt occurs. If a signal is delivered to the process during a wait, the application is responsible for dealing with the premature awakening in a reasonable manner.

When this function is being used to wait for interrupts, it can be woken up by a signal before an interrupt occurs and an interrupt may be missed if signals are delivered rapidly enough or at inopportune times. To decrease the chances of this, it is strongly suggested that you 1) do not use signals or 2) minimize their use in your application.

This function disables all interrupts for a very brief time to obtain accurate status information. If you call the function repeatedly in a loop (such as when busy-waiting for an interrupt to occur), this can interfere with system interrupts. It is strongly suggested that you do not busy-wait for interrupts.

Referenced by `main()`.

#### 4.20.2.4 DM7820\_Error DM7820\_General\_Get\_Version\_Info ( `DM7820_Board_Descriptor` \* *handle*, `uint8_t` \* *fpga\_type\_id*, `uint8_t` \* *fpga\_version*, `uint16_t` \* *svn\_version* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure.

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

Referenced by `main()`.

#### 4.20.2.5 DM7820\_Error DM7820\_General\_Is\_PCI\_Master ( DM7820\_Board\_Descriptor \* *handle*, uint8\_t \* *pci\_master* )

Determine whether or not a device is PCI master capable.

##### Parameters

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

##### Return values

0	Success.
-1	Failure.

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

Referenced by main().

#### 4.20.2.6 DM7820\_Error DM7820\_General\_Open\_Board ( uint8\_t *dev\_num*, DM7820\_Board\_Descriptor \*\* *handle* )

Open a DM7820 device file.

##### Parameters

<i>dev_num</i>	Minor number of DM7820 device file.
<i>handle</i>	Address where address of memory allocated for library device descriptor should be stored. If the first open of a device file fails, then NULL will be stored here.

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EBUSY The DM7820 device file with minor number <i>dev_num</i> is already open.</li> <li>ENODEV <i>dev_num</i> is not a valid DM7820 minor number; 2.4 kernel only.</li> <li>ENOMEM Library device descriptor memory allocation failed.</li> <li>ENXIO <i>dev_num</i> is not a valid DM7820 minor number; 2.6 kernel only.</li> </ul> Please see the open(2) man page for information on other possible values errno may have in this case.

##### Note

Once a device file is open, it cannot be opened again until it is closed.

When processing the open request, the driver disables & clears all device interrupts, enables PLX PCI interrupts, enables PLX local interrupt input, and enables PLX DMA channel 0/1 interrupts.

Referenced by main().

#### 4.20.2.7 DM7820\_Error DM7820\_General\_RemoveISR ( DM7820\_Board\_Descriptor \* *handle* )

Uninstall userspace ISR.



## Parameters

<i>handle</i>	Address of the device's library board descriptor.
---------------	---

## Return values

0	Success
-1	Failure

Referenced by main().

**4.20.2.8 DM7820\_Error DM7820\_General\_Reset ( DM7820\_Board\_Descriptor \* *handle* )**

Reset a DM7820 device.

## Parameters

<i>handle</i>	Address of device's library board descriptor.
---------------	---

## Return values

0	Success.
-1	Failure.

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

## Note

This function does not reset the PLX chip.

Referenced by main().

**4.20.2.9 DM7820\_Error DM7820\_General\_SetISRPriority ( DM7820\_Board\_Descriptor \* *handle*, int *priority* )**

Changes the Priority for the ISR thread.

## Parameters

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

## Return values

0	Success
---	---------

Referenced by main().

## 4.21 DM7820 user library incremental encoder functions

### Functions

- **DM7820\_Error DM7820\_IncEnc\_Configure** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **dm7820\_incenc\_phase\_filter** phase\_filter, **dm7820\_incenc\_input\_mode** input\_mode, **uint8\_t** enable\_input\_filter, **dm7820\_incenc\_channel\_mode** channel\_mode, **uint8\_t** enable\_index)  
*Configure the given incremental encoder.*
- **DM7820\_Error DM7820\_IncEnc\_Enable** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **uint8\_t** enable)  
*Enable or disable the given incremental encoder.*
- **DM7820\_Error DM7820\_IncEnc\_Enable\_Hold** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **uint8\_t** enable)  
*Enable or disable value register hold for the given incremental encoder.*
- **DM7820\_Error DM7820\_IncEnc\_Get\_Independent\_Value** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **dm7820\_incenc\_channel** channel, **uint16\_t** \*value)  
*Get 16-bit counter value of the given independent incremental encoder channel.*
- **DM7820\_Error DM7820\_IncEnc\_Get\_Joined\_Value** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **uint32\_t** \*value)  
*Get 32-bit counter value of the given independent incremental encoder whose channels are joined.*
- **DM7820\_Error DM7820\_IncEnc\_Get\_Status** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **dm7820\_incenc\_status\_condition** condition, **uint8\_t** \*occurred)  
*Determine whether or not the specified status condition has occurred for the given incremental encoder.*
- **DM7820\_Error DM7820\_IncEnc\_Set\_Independent\_Value** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **dm7820\_incenc\_channel** channel, **uint16\_t** value)  
*Set 16-bit counter value for the given independent incremental encoder channel.*
- **DM7820\_Error DM7820\_IncEnc\_Set\_Joined\_Value** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **uint32\_t** value)  
*Set 32-bit counter value for the given incremental encoder whose channels are joined.*
- **DM7820\_Error DM7820\_IncEnc\_Set\_Master** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_incenc\_encoder** encoder, **dm7820\_incenc\_master\_clock** master)  
*Set the master clock for the given incremental encoder.*

### 4.21.1 Detailed Description

DM7820\_Library\_General\_Functions

### 4.21.2 Function Documentation

- 4.21.2.1 **DM7820\_Error DM7820\_IncEnc.Configure** ( **DM7820\_Board\_Descriptor** \* handle, **dm7820\_incenc\_encoder** encoder, **dm7820\_incenc\_phase\_filter** phase\_filter, **dm7820\_incenc\_input\_mode** input\_mode, **uint8\_t** enable\_input\_filter, **dm7820\_incenc\_channel\_mode** channel\_mode, **uint8\_t** enable\_index )

Configure the given incremental encoder.

#### Parameters

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

The DM7820\_INCENC\_DISABLE\_PHASE\_FILTER\_TRANSITION, DM7820\_INCENC\_ENABLE\_PHASE\_FILTER\_TRANSITION, and DM7820\_INCENC\_RESET\_PHASE\_FILTER macros should be used modify transitions in

the phase filter.

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL encoder is not valid.</li> <li>• EINVAL input_mode is not valid.</li> <li>• EINVAL channel_mode is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

#### Note

The incremental encoder should be disabled before calling this function.

Referenced by do\_incenc(), and main().

#### 4.21.2.2 DM7820\_Error DM7820\_IncEnc\_Enable ( DM7820\_Board\_Descriptor \* handle, dm7820\_incenc\_encoder encoder, uint8\_t enable )

Enable or disable the given incremental encoder.

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL encoder is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do\_incenc(), and main().

#### 4.21.2.3 **DM7820\_Error** DM7820\_IncEnc.Enable\_Hold ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_incenc\_encoder** *encoder*, **uint8\_t** *enable* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL encoder is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do\_incenc(), and main().

#### 4.21.2.4 **DM7820\_Error** DM7820\_IncEnc.Get\_Independent\_Value ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_incenc\_encoder** *encoder*, **dm7820\_incenc\_channel** *channel*, **uint16\_t** \* *value* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL encoder is not valid.</li> <li>• EINVAL channel is not valid.</li> <li>• EOPNOTSUPP Incremental encoder channels A and B are joined.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

#### 4.21.2.5 **DM7820\_Error** DM7820\_IncEnc.Get\_Joined\_Value ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_incenc\_encoder** *encoder*, **uint32\_t** \* *value* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL encoder is not valid.</li> <li>• EINVAL channel is not valid.</li> <li>• EOPNOTSUPP Incremental encoder channels A and B are independent.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

#### 4.21.2.6 DM7820\_Error DM7820\_IncEnc.Get\_Status ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_incenc\_encoder *encoder*, dm7820\_incenc\_status\_condition *condition*, uint8\_t \* *occurred* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL encoder is not valid.</li> <li>• EINVAL condition is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

## Warning

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

## Note

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

Referenced by main().

#### 4.21.2.7 **DM7820\_Error** DM7820\_IncEnc.Set\_Independent\_Value ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_incenc\_encoder** *encoder*, **dm7820\_incenc\_channel** *channel*, **uint16\_t** *value* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL encoder is not valid.</li> <li>EINVAL channel is not valid.</li> <li>EOPNOTSUPP Incremental encoder channels A and B are joined.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do\_incenc(), and main().

#### 4.21.2.8 **DM7820\_Error** DM7820\_IncEnc.Set\_Joined\_Value ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_incenc\_encoder** *encoder*, **uint32\_t** *value* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL encoder is not valid.</li> <li>EINVAL channel is not valid.</li> <li>EOPNOTSUPP Incremental encoder channels A and B are independent.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

#### 4.21.2.9 **DM7820\_Error** DM7820\_IncEnc.Set\_Master ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_incenc\_encoder** *encoder*, **dm7820\_incenc\_master\_clock** *master* )

Set the master clock for the given incremental encoder.

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"><li>• <code>EINVAL</code> encoder is not valid.</li><li>• <code>EINVAL</code> master is not valid.</li><li>• <code>EOPNOTSUPP</code> master is equal to <code>DM7820_INCENC_MASTER_RESERVED</code>.</li></ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

## Note

The incremental encoder should be disabled before calling this function.

Referenced by `do_incenc()`, and `main()`.

## 4.22 DM7820 user library pulse width modulator functions

### Functions

- [DM7820\\_Error DM7820\\_PWM\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [uint8\\_t](#) enable)  
*Enable or disable the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Period](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [uint32\\_t](#) period)  
*Set the period for the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Period\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [dm7820\\_pwm\\_period\\_master\\_clock](#) master)  
*Set the period master clock for the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Width](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [dm7820\\_pwm\\_output](#) output, [uint16\\_t](#) width)  
*Set the width for the specified output on the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Width\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [dm7820\\_pwm\\_width\\_master\\_clock](#) master)  
*Set the width master clock for the given pulse width modulator (PWM).*

### 4.22.1 Detailed Description

DM7820\_Library\_IncEnc\_Functions

### 4.22.2 Function Documentation

#### 4.22.2.1 [DM7820\\_Error DM7820\\_PWM\\_Enable](#) ( [DM7820\\_Board\\_Descriptor](#) \* handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [uint8\\_t](#) enable )

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

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> pwm is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `do_pwm()`, and `main()`.

#### 4.22.2.2 [DM7820\\_Error DM7820\\_PWM.Set\\_Period](#) ( [DM7820\\_Board\\_Descriptor](#) \* handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [uint32\\_t](#) period )

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



## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>pwm</i> is not valid.</li> <li>ERANGE <i>period</i> is greater than 0x10000.</li> <li>ERANGE <i>period</i> is equal to 0.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

## Note

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

Referenced by do\_pwm(), and main().

#### 4.22.2.3 DM7820\_Error DM7820\_PWM\_Set\_Period\_Master ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_pwm\_modulator *pwm*, dm7820\_pwm\_period\_master\_clock *master* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>pwm</i> is not valid.</li> <li>EINVAL <i>master</i> is not valid.</li> <li>EOPNOTSUPP <i>master</i> is equal to DM7820_PWM_PERIOD_MASTER_RESERVED.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

## Note

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

Referenced by `do_pwm()`, and `main()`.

#### 4.22.2.4 DM7820\_Error DM7820\_PWM.Set.Width ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_pwm\_modulator *pwm*, dm7820\_pwm\_output *output*, uint16\_t *width* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>pwm</i> is not valid.</li> <li>EINVAL <i>output</i> is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

Referenced by `do_pwm()`, and `main()`.

#### 4.22.2.5 DM7820\_Error DM7820\_PWM.Set.Width.Master ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_pwm\_modulator *pwm*, dm7820\_pwm\_width\_master\_clock *master* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL <i>pwm</i> is not valid.</li> <li>EINVAL <i>master</i> is not valid.</li> <li>EOPNOTSUPP <i>master</i> is equal to <code>DM7820_PWM_WIDTH_MASTER_RESERVED</code>.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

**Note**

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

Referenced by `do_pwm()`, and `main()`.

## 4.23 DM7820 user library programmable clock functions

### Functions

- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_master\\_clock](#) master)  
Select the master clock for the given programmable clock.
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_mode](#) mode)  
Select the mode for the given programmable clock.
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Period](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [uint32\\_t](#) period)  
Set the period for the given programmable clock.
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Start\\_Trigger](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_ - clock](#) clock, [dm7820\\_prgclk\\_start\\_trigger](#) start)  
Set the start trigger for the given programmable clock.
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Stop\\_Trigger](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_ - clock](#) clock, [dm7820\\_prgclk\\_stop\\_trigger](#) stop)  
Set the stop trigger for the given programmable clock.

### 4.23.1 Detailed Description

DM7820\_Library\_PWM\_Functions

### 4.23.2 Function Documentation

#### 4.23.2.1 [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Master](#) ( [DM7820\\_Board\\_Descriptor](#) \* handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_master\\_clock](#) master )

Select the master clock for the given programmable clock.

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL clock is not valid.</li> <li>• EINVAL master is not valid.</li> <li>• EOPNOTSUPP master is equal to DM7820_PRGCLK_MASTER_RESERVED.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

#### Note

The clock should be disabled before calling this function.

Referenced by `main()`.

#### 4.23.2.2 DM7820\_Error DM7820\_PrgClk\_Set\_Mode ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_prgclk\_clock *clock*, dm7820\_prgclk\_mode *mode* )

Select the mode for the given programmable clock.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL clock is not valid.</li> <li>EINVAL mode is not valid.</li> <li>EOPNOTSUPP mode is equal to DM7820_PRGCLK_MODE_RESERVED.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

##### Note

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

Referenced by clean\_up(), and main().

#### 4.23.2.3 DM7820\_Error DM7820\_PrgClk\_Set\_Period ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_prgclk\_clock *clock*, uint32\_t *period* )

Set the period for the given programmable clock.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL clock is not valid.</li> <li>ERANGE period is greater than 0x10000.</li> <li>ERANGE period is equal to 0.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

#### 4.23.2.4 DM7820\_Error DM7820\_PrgClk\_Set\_Start\_Trigger ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_prgclk\_clock *clock*, dm7820\_prgclk\_start\_trigger *start* )

Set the start trigger for the given programmable clock.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL clock is not valid.</li> <li>• EINVAL start is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

##### Note

The clock should be disabled before calling this function.

Referenced by main().

#### 4.23.2.5 DM7820\_Error DM7820\_PrgClk\_Set\_Stop\_Trigger ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_prgclk\_clock *clock*, dm7820\_prgclk\_stop\_trigger *stop* )

Set the stop trigger for the given programmable clock.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• EINVAL clock is not valid.</li> <li>• EINVAL stop is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

##### Note

The clock should be disabled before calling this function.

Referenced by main().

## 4.24 DM7820 user library standard I/O functions

### Functions

- **DM7820\_Error DM7820\_StdIO\_Get\_Input** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Port** port, **uint16\_t** \*value)  
*Read a value from the given standard I/O port.*
- **DM7820\_Error DM7820\_StdIO\_Set\_IO\_Mode** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Port** port, **uint16\_t** bits, **DM7820\_StdIO\_IO\_Mode** mode)  
*Set the mode for specific bits in the given standard I/O port.*
- **DM7820\_Error DM7820\_StdIO\_Set\_Output** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Port** port, **uint16\_t** value)  
*Write a value to the given standard I/O port.*
- **DM7820\_Error DM7820\_StdIO\_Set\_Periph\_Mode** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Port** port, **uint16\_t** bits, **DM7820\_StdIO\_Periph\_Mode** mode)  
*Set the peripheral output mode for specific bits in the given standard I/O port.*
- **DM7820\_Error DM7820\_StdIO\_Strobe\_Input** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Strobe** strobe, **uint8\_t** \*state)  
*Determine state of given strobe signal.*
- **DM7820\_Error DM7820\_StdIO\_Strobe\_Mode** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Strobe** strobe, **uint8\_t** output)  
*Set the direction (input or output) for the given strobe signal.*
- **DM7820\_Error DM7820\_StdIO\_Strobe\_Output** (**DM7820\_Board\_Descriptor** \*handle, **DM7820\_StdIO\_Strobe** strobe, **uint8\_t** state)  
*Set state of given strobe signal.*

### 4.24.1 Detailed Description

DM7820\_Library\_PrgClk\_Functions

### 4.24.2 Function Documentation

#### 4.24.2.1 **DM7820\_Error DM7820\_StdIO\_Get\_Input** ( **DM7820\_Board\_Descriptor** \* handle, **DM7820\_StdIO\_Port** port, **uint16\_t** \* value )

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

#### Parameters

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

#### Note

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> port is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values <code>errno</code> may have in this case.

Referenced by `main()`.

#### 4.24.2.2 **DM7820\_Error** DM7820\_StdIO\_Set\_IO\_Mode ( **DM7820\_Board\_Descriptor** \* *handle*, **DM7820\_StdIO\_Port** *port*, **uint16\_t** *bits*, **DM7820\_StdIO\_IO\_Mode** *mode* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL port is not valid.</li> <li>EINVAL mode is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `do_digital_io()`, and `main()`.

#### 4.24.2.3 **DM7820\_Error** DM7820\_StdIO\_Set\_Output ( **DM7820\_Board\_Descriptor** \* *handle*, **DM7820\_StdIO\_Port** *port*, **uint16\_t** *value* )

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

##### Parameters

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

##### Note

Any port bit not set to output is ignored.

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL port is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `main()`.



#### 4.24.2.4 **DM7820\_Error** DM7820\_StdIO\_Set\_Periph\_Mode ( DM7820\_Board\_Descriptor \* *handle*, DM7820\_StdIO\_Port *port*, uint16\_t *bits*, DM7820\_StdIO\_Periph\_Mode *mode* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL port is not valid.</li> <li>EINVAL mode is not valid.</li> <li>EOPNOTSUPP port does not support mode.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do\_digital\_io(), and main().

#### 4.24.2.5 **DM7820\_Error** DM7820\_StdIO\_Strobe\_Input ( DM7820\_Board\_Descriptor \* *handle*, DM7820\_StdIO\_Strobe *strobe*, uint8\_t \* *state* )

Determine state of given strobe signal.

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL strobe is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by main().

#### 4.24.2.6 **DM7820\_Error** DM7820\_StdIO\_Strobe\_Mode ( DM7820\_Board\_Descriptor \* *handle*, DM7820\_StdIO\_Strobe *strobe*, uint8\_t *output* )

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

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> strobe is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `main()`.

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

Set state of given strobe signal.

## Parameters

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

## Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> strobe is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `main()`.

## 4.25 DM7820 user library 8254 timer/counter functions

### Functions

- **DM7820\_Error DM7820\_TmrCtr\_Get\_Status** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_tmrctr\_timer** timer, **uint8\_t** \*occurred)  
Determine whether or not a status condition has occurred for the given timer/counter.
- **DM7820\_Error DM7820\_TmrCtr\_Program** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_tmrctr\_timer** timer, **dm7820\_tmrctr\_waveform** waveform, **dm7820\_tmrctr\_count\_mode** count\_mode, **uint16\_t** divisor)  
Program the given 8254 timer/counter. This will 1) set the waveform mode, 2) set the count mode, and 3) load a divisor into the timer.
- **DM7820\_Error DM7820\_TmrCtr\_Read** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_tmrctr\_timer** timer, **uint16\_t** \*value)  
Read the value of the given 8254 timer/counter.
- **DM7820\_Error DM7820\_TmrCtr\_Select\_Clock** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_tmrctr\_timer** timer, **dm7820\_tmrctr\_clock** clock)  
Select the clock input for the given 8254 timer/counter.
- **DM7820\_Error DM7820\_TmrCtr\_Select\_Gate** (**DM7820\_Board\_Descriptor** \*handle, **dm7820\_tmrctr\_timer** timer, **dm7820\_tmrctr\_gate** gate)  
Select the gate input for the given 8254 timer/counter.

### 4.25.1 Detailed Description

DM7820\_Library\_StdIO\_Functions

### 4.25.2 Function Documentation

#### 4.25.2.1 **DM7820\_Error DM7820\_TmrCtr\_Get\_Status** ( **DM7820\_Board\_Descriptor** \* handle, **dm7820\_tmrctr\_timer** timer, **uint8\_t** \* occurred )

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

#### Parameters

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

#### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>• <code>EINVAL</code> timer is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

#### Warning

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

**Note**

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

Referenced by `main()`.

#### 4.25.2.2 **DM7820\_Error** DM7820\_TmrCtr\_Program ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_tmrctr\_timer** *timer*, **dm7820\_tmrctr\_waveform** *waveform*, **dm7820\_tmrctr\_count\_mode** *count\_mode*, **uint16\_t** *divisor* )

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

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL timer is not valid.</li> <li>EINVAL waveform is not valid.</li> <li>EINVAL count_mode is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

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

#### 4.25.2.3 **DM7820\_Error** DM7820\_TmrCtr\_Read ( **DM7820\_Board\_Descriptor** \* *handle*, **dm7820\_tmrctr\_timer** *timer*, **uint16\_t** \* *value* )

Read the value of the given 8254 timer/counter.

**Parameters**

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

**Return values**

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL timer is not valid.</li> </ul> Please see the <code>ioctl(2)</code> man page for information on other possible values errno may have in this case.

Referenced by `main()`.

#### 4.25.2.4 DM7820\_Error DM7820\_TmrCtr\_Select\_Clock ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_tmrctr\_timer *timer*, dm7820\_tmrctr\_clock *clock* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL timer is not valid.</li> <li>EINVAL clock is not valid.</li> <li>EOPNOTSUPP clock is equal to DM7820_TMRCTR_CLOCK_RESERVED.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by do\_8254(), and main().

#### 4.25.2.5 DM7820\_Error DM7820\_TmrCtr\_Select\_Gate ( DM7820\_Board\_Descriptor \* *handle*, dm7820\_tmrctr\_timer *timer*, dm7820\_tmrctr\_gate *gate* )

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

##### Parameters

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

##### Return values

0	Success.
-1	Failure. errno may be set as follows: <ul style="list-style-type: none"> <li>EINVAL timer is not valid.</li> <li>EINVAL gate is not valid.</li> </ul> Please see the ioctl(2) man page for information on other possible values errno may have in this case.

Referenced by disable\_timers(), do\_8254(), and main().

## 4.26 DM7820 register header file

### Modules

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

### 4.26.1 Detailed Description

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

### Macros

- #define [DM7820\\_BAR0\\_INTCSR](#) 0x68  
*PLX interrupt control/status.*
- #define [DM7820\\_BAR0\\_DMAMODE0](#) 0x80  
*PLX DMA channel 0 mode.*
- #define [DM7820\\_BAR0\\_DMAPADR0](#) 0x84  
*PLX DMA channel 0 PCI address.*
- #define [DM7820\\_BAR0\\_DMALADR0](#) 0x88  
*PLX DMA channel 0 local address.*
- #define [DM7820\\_BAR0\\_DMASIZ0](#) 0x8C  
*PLX DMA channel 0 transfer size.*
- #define [DM7820\\_BAR0\\_DMADPR0](#) 0x90  
*PLX DMA channel 0 descriptor pointer.*
- #define [DM7820\\_BAR0\\_DMAMODE1](#) 0x94  
*PLX DMA channel 1 mode.*
- #define [DM7820\\_BAR0\\_DMAPADR1](#) 0x98  
*PLX DMA channel 1 PCI address.*
- #define [DM7820\\_BAR0\\_DMALADR1](#) 0x9C  
*PLX DMA channel 1 local address.*
- #define [DM7820\\_BAR0\\_DMASIZ1](#) 0xA0  
*PLX DMA channel 1 transfer size.*
- #define [DM7820\\_BAR0\\_DMADPR1](#) 0xA4  
*PLX DMA channel 1 descriptor pointer.*
- #define [DM7820\\_BAR0\\_DMACSR0](#) 0xA8  
*PLX DMA channel 0 command/status.*
- #define [DM7820\\_BAR0\\_DMACSR1](#) 0xA9  
*PLX DMA channel 1 command/status.*
- #define [DM7820\\_BAR0\\_DMAARB](#) 0xAC  
*PLX DMA arbitration.*
- #define [DM7820\\_BAR0\\_DMATHR](#) 0xB0  
*PLX DMA threshold.*
- #define [DM7820\\_BAR0\\_DMADA0](#) 0xB4  
*PLX DMA channel 0 PCI dual address cycles upper address.*
- #define [DM7820\\_BAR0\\_DMADA1](#) 0xB8  
*PLX DMA channel 1 PCI dual address cycles upper address.*

### 4.27.1 Detailed Description

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

### Macros

- #define [DM7820\\_BAR1\\_INTCSR](#) 0x68  
*PLX interrupt control/status.*
- #define [DM7820\\_BAR1\\_DMAMODE0](#) 0x80  
*PLX DMA channel 0 mode.*
- #define [DM7820\\_BAR1\\_DMAPADR0](#) 0x84  
*PLX DMA channel 0 PCI address.*
- #define [DM7820\\_BAR1\\_DMALADR0](#) 0x88  
*PLX DMA channel 0 local address.*
- #define [DM7820\\_BAR1\\_DMASIZ0](#) 0x8C  
*PLX DMA channel 0 transfer size.*
- #define [DM7820\\_BAR1\\_DMADPR0](#) 0x90  
*PLX DMA channel 0 descriptor pointer.*
- #define [DM7820\\_BAR1\\_DMAMODE1](#) 0x94  
*PLX DMA channel 1 mode.*
- #define [DM7820\\_BAR1\\_DMAPADR1](#) 0x98  
*PLX DMA channel 1 PCI address.*
- #define [DM7820\\_BAR1\\_DMALADR1](#) 0x9C  
*PLX DMA channel 1 local address.*
- #define [DM7820\\_BAR1\\_DMASIZ1](#) 0xA0  
*PLX DMA channel 1 transfer size.*
- #define [DM7820\\_BAR1\\_DMADPR1](#) 0xA4  
*PLX DMA channel 1 descriptor pointer.*
- #define [DM7820\\_BAR1\\_DMACSR0](#) 0xA8  
*PLX DMA channel 0 command/status.*
- #define [DM7820\\_BAR1\\_DMACSR1](#) 0xA9  
*PLX DMA channel 1 command/status.*
- #define [DM7820\\_BAR1\\_DMAARB](#) 0xAC  
*PLX DMA arbitration.*
- #define [DM7820\\_BAR1\\_DMATHR](#) 0xB0  
*PLX DMA threshold.*
- #define [DM7820\\_BAR1\\_DMADA0](#) 0xB4  
*PLX DMA channel 0 PCI dual address cycles upper address.*
- #define [DM7820\\_BAR1\\_DMADA1](#) 0xB8  
*PLX DMA channel 1 PCI dual address cycles upper address.*

### 4.28.1 Detailed Description

DM7820\_Register\_BAR0\_Offsets



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

### Modules

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

### Macros

- [#define PLX9056\\_DMA\\_WIDTH\\_MASK 0x00000003](#)
- [#define PLX9056\\_DMA\\_WIDTH\\_8 0x00000000](#)
- [#define PLX9056\\_DMA\\_WIDTH\\_16 0x00000001](#)
- [#define PLX9056\\_DMA\\_WIDTH\\_32 0x00000002](#)
- [#define PLX9056\\_DMA\\_WAITSTATES\\_MASK 0x0000003c](#)
- [#define PLX9056\\_DMA\\_READY 0x00000040](#)
- [#define PLX9056\\_DMA\\_BURST 0x00000080](#)
- [#define PLX9056\\_DMA\\_LOCAL\\_BURST 0x00000100](#)
- [#define PLX9056\\_DMA\\_SCATTERGATHER 0x00000200](#)
- [#define PLX9056\\_DMA\\_DONE\\_INTERRUPT 0x00000400](#)
- [#define PLX9056\\_DMA\\_LOCAL\\_ADDRESSING\\_MODE 0x00000800](#)
- [#define PLX9056\\_DMA\\_DEMAND\\_MODE 0x00001000](#)
- [#define PLX9056\\_DMA\\_MEMWRITE\\_INV 0x00002000](#)
- [#define PLX9056\\_DMA\\_EOT\\_ENABLE 0x00004000](#)
- [#define PLX9056\\_DMA\\_FAST\\_SLOW\\_TERM 0x00008000](#)
- [#define PLX9056\\_DMA\\_CLEAR\\_COUNT 0x00010000](#)
- [#define PLX9056\\_DMA\\_INTERRUPT\\_SEL 0x00020000](#)
- [#define PLX9056\\_DMA\\_DAC\\_CHAIN 0x00040000](#)
- [#define PLX9056\\_DMA\\_EOT\\_END 0x00080000](#)
- [#define PLX9056\\_DMA\\_RING\\_MODE 0x00100000](#)
- [#define PLX9056\\_DMA\\_RING\\_CONTROL 0x00200000](#)

### 4.29.1 Detailed Description

DM7820\_Register\_BAR1\_Offsets

## 4.29.2 Macro Definition Documentation

### 4.29.2.1 `#define PLX9056_DMA_BURST 0x00000080`

Continuous burst enable (use as mask and control)

Definition at line 560 of file dm7820\_registers.h.

### 4.29.2.2 `#define PLX9056_DMA_CLEAR_COUNT 0x00010000`

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

Definition at line 629 of file dm7820\_registers.h.

### 4.29.2.3 `#define PLX9056_DMA_DAC_CHAIN 0x00040000`

DAC Chain load (not supported most chipsets)

Definition at line 643 of file dm7820\_registers.h.

### 4.29.2.4 `#define PLX9056_DMA_DEMAND_MODE 0x00001000`

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

Definition at line 601 of file dm7820\_registers.h.

### 4.29.2.5 `#define PLX9056_DMA_DONE_INTERRUPT 0x00000400`

DMA done interrupt enable (use as mask and control) At the time the DMA cycle is completed an interrupt will occur on the PCI bus if the master interrupt is enabled IF PLX9056\_DMA\_CLEAR\_COUNT is enabled then the interrupt will not occur until the byte count is cleared in the PCI memory table (scatter/gather mode is enabled in this case).

Definition at line 585 of file dm7820\_registers.h.

### 4.29.2.6 `#define PLX9056_DMA_EOT_ENABLE 0x00004000`

EOT# Enable

Definition at line 613 of file dm7820\_registers.h.

### 4.29.2.7 `#define PLX9056_DMA_EOT_END 0x00080000`

EOT# End Link

Definition at line 649 of file dm7820\_registers.h.

### 4.29.2.8 `#define PLX9056_DMA_FAST_SLOW_TERM 0x00008000`

Fast/Slow Terminate mode selected

Definition at line 619 of file dm7820\_registers.h.

**4.29.2.9 #define PLX9056\_DMA\_INTERRUPT\_SEL 0x00020000**

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

Definition at line 637 of file dm7820\_registers.h.

**4.29.2.10 #define PLX9056\_DMA\_LOCAL\_ADDRESSING\_MODE 0x00000800**

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

Definition at line 593 of file dm7820\_registers.h.

**4.29.2.11 #define PLX9056\_DMA\_LOCAL\_BURST 0x00000100**

Lecal burst enable (use as mask and control)

Definition at line 566 of file dm7820\_registers.h.

**4.29.2.12 #define PLX9056\_DMA\_MEMWRITE\_INV 0x00002000**

Memory write and invalidate mode.

Definition at line 607 of file dm7820\_registers.h.

**4.29.2.13 #define PLX9056\_DMA\_READY 0x00000040**

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

Definition at line 554 of file dm7820\_registers.h.

**4.29.2.14 #define PLX9056\_DMA\_RING\_CONTROL 0x00200000**

Ring Management Void Stop Control Valid when PLX9056\_DMA\_RING\_MODE is set When this bit is 0 the Scatter/-Gather controller will load the descriptor and execute the DMA. If this bit is 1 then the Scatter/Gather controller will stop polling when the valid bit goes 0. The host will need to restart the DMA controller by setting the DMACSRx[1] to 1

Definition at line 672 of file dm7820\_registers.h.

**4.29.2.15 #define PLX9056\_DMA\_RING\_MODE 0x00100000**

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

Definition at line 660 of file dm7820\_registers.h.

**4.29.2.16 #define PLX9056\_DMA\_SCATTERGATHER 0x00000200**

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

Definition at line 574 of file dm7820\_registers.h.

4.29.2.17 `#define PLX9056_DMA_WAITSTATES_MASK 0x0000003c`

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

Definition at line 548 of file dm7820\_registers.h.

4.29.2.18 `#define PLX9056_DMA_WIDTH_16 0x00000001`

Local Bus Data Width 16 Bits.

Definition at line 535 of file dm7820\_registers.h.

4.29.2.19 `#define PLX9056_DMA_WIDTH_32 0x00000002`

Local Bus Data Width 32 Bits.

Definition at line 541 of file dm7820\_registers.h.

4.29.2.20 `#define PLX9056_DMA_WIDTH_8 0x00000000`

Local Bus Data Width 8 Bits.

Definition at line 529 of file dm7820\_registers.h.

4.29.2.21 `#define PLX9056_DMA_WIDTH_MASK 0x00000003`

DM7820\_Register\_StdIO Local Bus Data Width Mask

Definition at line 523 of file dm7820\_registers.h.

## 4.30 DM7820 register board control offsets

### Macros

- #define [DM7820\\_BAR2\\_FPGA\\_VERSION](#) 0x0000  
*FPGA version and type identifiers.*
- #define [DM7820\\_BAR2\\_SVN\\_VERSION](#) 0x0002  
*FPGA source code revision control version identifier.*
- #define [DM7820\\_BAR2\\_BOARD\\_RESET](#) 0x0004  
*Board reset.*
- #define [DM7820\\_BAR2\\_BRD\\_STAT](#) 0x0008  
*Board status.*
- #define [DM7820\\_BAR2\\_INTERRUPT\\_ENABLE](#) 0x0010  
*Local interrupt enable.*
- #define [DM7820\\_BAR2\\_INTERRUPT\\_STATUS](#) 0x0012  
*Local interrupt status.*

### 4.30.1 Detailed Description

## 4.31 DM7820 register standard I/O offsets

### Macros

- #define [DM7820\\_BAR2\\_PORT0\\_OUTPUT](#) 0x0040  
*Port 0 output value.*
- #define [DM7820\\_BAR2\\_PORT0\\_INPUT](#) 0x0042  
*Port 0 input value.*
- #define [DM7820\\_BAR2\\_PORT0\\_TRISTATE](#) 0x0044  
*Port 0 direction.*
- #define [DM7820\\_BAR2\\_PORT0\\_MODE](#) 0x0046  
*Port 0 operating mode.*
- #define [DM7820\\_BAR2\\_PORT1\\_OUTPUT](#) 0x0048  
*Port 1 output value.*
- #define [DM7820\\_BAR2\\_PORT1\\_INPUT](#) 0x004A  
*Port 1 input value.*
- #define [DM7820\\_BAR2\\_PORT1\\_TRISTATE](#) 0x004C  
*Port 1 direction.*
- #define [DM7820\\_BAR2\\_PORT1\\_MODE](#) 0x004E  
*Port 1 operating mode.*
- #define [DM7820\\_BAR2\\_PORT2\\_OUTPUT](#) 0x0050  
*Port 2 output value.*
- #define [DM7820\\_BAR2\\_PORT2\\_INPUT](#) 0x0052  
*Port 2 input value.*
- #define [DM7820\\_BAR2\\_PORT2\\_TRISTATE](#) 0x0054  
*Port 2 direction.*
- #define [DM7820\\_BAR2\\_PORT2\\_MODE](#) 0x0056  
*Port 2 operating mode.*
- #define [DM7820\\_BAR2\\_STROBE\\_STATUS](#) 0x0058  
*Port 2 strobe signal status.*
- #define [DM7820\\_BAR2\\_PORT0\\_PERIPH\\_SEL\\_L](#) 0x0060  
*Port 0 bits 0-7 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT0\\_PERIPH\\_SEL\\_H](#) 0x0062  
*Port 0 bits 8-15 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT1\\_PERIPH\\_SEL\\_L](#) 0x0064  
*Port 1 bits 0-7 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT1\\_PERIPH\\_SEL\\_H](#) 0x0066  
*Port 1 bits 8-15 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT2\\_PERIPH\\_SEL\\_L](#) 0x0068  
*Port 2 bits 0-7 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT2\\_PERIPH\\_SEL\\_H](#) 0x006A  
*Port 2 bits 8-15 peripheral select.*

### 4.31.1 Detailed Description

DM7820\_Register\_General

## 4.32 DM7820 register 8254 timer/counter offsets

### Macros

- #define [DM7820\\_BAR2\\_TC\\_ID](#) 0x0080  
*8254 timer/counter block identifier*
- #define [DM7820\\_BAR2\\_TC\\_INT](#) 0x0082  
*8254 timer/counter interrupt control/status*
- #define [DM7820\\_BAR2\\_TC\\_A0\\_CONTROL](#) 0x0084  
*8254 timer/counter A0 control*
- #define [DM7820\\_BAR2\\_TC\\_A1\\_CONTROL](#) 0x0086  
*8254 timer/counter A1 control*
- #define [DM7820\\_BAR2\\_TC\\_A2\\_CONTROL](#) 0x0088  
*8254 timer/counter A2 control*
- #define [DM7820\\_BAR2\\_TC\\_B0\\_CONTROL](#) 0x008A  
*8254 timer/counter B0 control*
- #define [DM7820\\_BAR2\\_TC\\_B1\\_CONTROL](#) 0x008C  
*8254 timer/counter B1 control*
- #define [DM7820\\_BAR2\\_TC\\_B2\\_CONTROL](#) 0x008E  
*8254 timer/counter B2 control*

### 4.32.1 Detailed Description

## 4.33 DM7820 register FIFO 0 offsets

### Macros

- #define [DM7820\\_BAR2\\_FIFO0\\_ID](#) 0x00C0  
*FIFO 0 block identifier.*
- #define [DM7820\\_BAR2\\_FIFO0\\_INT](#) 0x00C2  
*FIFO 0 interrupt control/status.*
- #define [DM7820\\_BAR2\\_FIFO0\\_IN\\_CLK](#) 0x00C4  
*FIFO 0 input clock.*
- #define [DM7820\\_BAR2\\_FIFO0\\_OUT\\_CLK](#) 0x00C6  
*FIFO 0 output clock.*
- #define [DM7820\\_BAR2\\_FIFO0\\_IN\\_DATA\\_DREQ](#) 0x00C8  
*FIFO 0 data input and PLX DMA request source.*
- #define [DM7820\\_BAR2\\_FIFO0\\_CON\\_STAT](#) 0x00CA  
*FIFO 0 control/status.*
- #define [DM7820\\_BAR2\\_FIFO0\\_RW\\_PORT](#) 0x00CC  
*FIFO 0 PCI read/write port.*

### 4.33.1 Detailed Description

DM7820\_Register\_TmrCtr



## 4.34 DM7820 register FIFO 1 offsets

### Macros

- #define `DM7820_BAR2_FIFO1_ID` 0x00D0  
*FIFO 1 block identifier.*
- #define `DM7820_BAR2_FIFO1_INT` 0x00D2  
*FIFO 1 interrupt control/status.*
- #define `DM7820_BAR2_FIFO1_IN_CLK` 0x00D4  
*FIFO 1 input clock.*
- #define `DM7820_BAR2_FIFO1_OUT_CLK` 0x00D6  
*FIFO 1 output clock.*
- #define `DM7820_BAR2_FIFO1_IN_DATA_DREQ` 0x00D8  
*FIFO 1 data input and PLX DMA request source.*
- #define `DM7820_BAR2_FIFO1_CON_STAT` 0x00DA  
*FIFO 1 control/status.*
- #define `DM7820_BAR2_FIFO1_RW_PORT` 0x00DC  
*FIFO 1 PCI read/write port.*

### 4.34.1 Detailed Description

DM7820\_Register\_FIFO\_0

## 4.35 DM7820 register programmable clock 0 offsets

### Macros

- #define `DM7820_BAR2_PRGCLK0_ID` 0x0100  
*Programmable clock 0 block identifier.*
- #define `DM7820_BAR2_PRGCLK0_MODE` 0x0102  
*Programmable clock 0 operating mode.*
- #define `DM7820_BAR2_PRGCLK0_CLK` 0x0104  
*Programmable clock 0 master clock.*
- #define `DM7820_BAR2_PRGCLK0_START_STOP` 0x0106  
*Programmable clock 0 start/stop triggers.*
- #define `DM7820_BAR2_PRGCLK0_PERIOD` 0x0108  
*Programmable clock 0 period.*

### 4.35.1 Detailed Description

DM7820\_Register\_FIFO\_1

## 4.36 DM7820 register programmable clock 1 offsets

### Macros

- #define `DM7820_BAR2_PRGCLK1_ID` 0x0140  
*Programmable clock 1 block identifier.*
- #define `DM7820_BAR2_PRGCLK1_MODE` 0x0142  
*Programmable clock 1 operating mode.*
- #define `DM7820_BAR2_PRGCLK1_CLK` 0x0144  
*Programmable clock 1 master clock.*
- #define `DM7820_BAR2_PRGCLK1_START_STOP` 0x0146  
*Programmable clock 1 start/stop triggers.*
- #define `DM7820_BAR2_PRGCLK1_PERIOD` 0x0148  
*Programmable clock 1 period.*

### 4.36.1 Detailed Description

DM7820\_Register\_PrgClk\_0

## 4.37 DM7820 register programmable clock 2 offsets

### Macros

- #define [DM7820\\_BAR2\\_PRGCLK2\\_ID](#) 0x0180  
*Programmable clock 2 block identifier.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_MODE](#) 0x0182  
*Programmable clock 2 operating mode.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_CLK](#) 0x0184  
*Programmable clock 2 master clock.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_START\\_STOP](#) 0x0186  
*Programmable clock 2 start/stop triggers.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_PERIOD](#) 0x0188  
*Programmable clock 2 period.*

### 4.37.1 Detailed Description

DM7820\_Register\_PrgClk\_1

## 4.38 DM7820 register programmable clock 3 offsets

### Macros

- #define `DM7820_BAR2_PRGCLK3_ID` 0x01C0  
*Programmable clock 3 block identifier.*
- #define `DM7820_BAR2_PRGCLK3_MODE` 0x01C2  
*Programmable clock 3 operating mode.*
- #define `DM7820_BAR2_PRGCLK3_CLK` 0x01C4  
*Programmable clock 3 master clock.*
- #define `DM7820_BAR2_PRGCLK3_START_STOP` 0x01C6  
*Programmable clock 3 start/stop triggers.*
- #define `DM7820_BAR2_PRGCLK3_PERIOD` 0x01C8  
*Programmable clock 3 period.*

### 4.38.1 Detailed Description

DM7820\_Register\_PrgClk\_2

## 4.39 DM7820 register advanced interrupt 0 offsets

### Macros

- #define [DM7820\\_BAR2\\_ADVINT0\\_ID](#) 0x0200  
*Advanced interrupt 0 block identifier.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_INT\\_MODE](#) 0x0202  
*Advanced interrupt 0 mode.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_CLK](#) 0x0204  
*Advanced interrupt 0 master clock.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT0\\_MASK](#) 0x0208  
*Advanced interrupt 0 standard I/O port 0 mask.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT1\\_MASK](#) 0x020A  
*Advanced interrupt 0 standard I/O port 1 mask.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT2\\_MASK](#) 0x020C  
*Advanced interrupt 0 standard I/O port 2 mask.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT0\\_CMP](#) 0x0210  
*Advanced interrupt 0 standard I/O port 0 event mode compare.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT1\\_CMP](#) 0x0212  
*Advanced interrupt 0 standard I/O port 1 event mode compare.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT2\\_CMP](#) 0x0214  
*Advanced interrupt 0 standard I/O port 2 event mode compare.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT0\\_CAPT](#) 0x0218  
*Advanced interrupt 0 standard I/O port 0 value capture.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT1\\_CAPT](#) 0x021A  
*Advanced interrupt 0 standard I/O port 1 value capture.*
- #define [DM7820\\_BAR2\\_ADVINT0\\_PORT2\\_CAPT](#) 0x021C  
*Advanced interrupt 0 standard I/O port 2 value capture.*

### 4.39.1 Detailed Description

DM7820\_Register\_PrgClk\_3

## 4.40 DM7820 register advanced interrupt 1 offsets

### Macros

- #define `DM7820_BAR2_ADVINT1_ID` 0x0240  
*Advanced interrupt 1 block identifier.*
- #define `DM7820_BAR2_ADVINT1_INT_MODE` 0x0242  
*Advanced interrupt 1 mode.*
- #define `DM7820_BAR2_ADVINT1_CLK` 0x0244  
*Advanced interrupt 1 master clock.*
- #define `DM7820_BAR2_ADVINT1_PORT0_MASK` 0x0248  
*Advanced interrupt 1 standard I/O port 0 mask.*
- #define `DM7820_BAR2_ADVINT1_PORT1_MASK` 0x024A  
*Advanced interrupt 1 standard I/O port 1 mask.*
- #define `DM7820_BAR2_ADVINT1_PORT2_MASK` 0x024C  
*Advanced interrupt 1 standard I/O port 2 mask.*
- #define `DM7820_BAR2_ADVINT1_PORT0_CMP` 0x0250  
*Advanced interrupt 1 standard I/O port 0 event mode compare.*
- #define `DM7820_BAR2_ADVINT1_PORT1_CMP` 0x0252  
*Advanced interrupt 1 standard I/O port 1 event mode compare.*
- #define `DM7820_BAR2_ADVINT1_PORT2_CMP` 0x0254  
*Advanced interrupt 1 standard I/O port 2 event mode compare.*
- #define `DM7820_BAR2_ADVINT1_PORT0_CAPT` 0x0258  
*Advanced interrupt 1 standard I/O port 0 value capture.*
- #define `DM7820_BAR2_ADVINT1_PORT1_CAPT` 0x025A  
*Advanced interrupt 1 standard I/O port 1 value capture.*
- #define `DM7820_BAR2_ADVINT1_PORT2_CAPT` 0x025C  
*Advanced interrupt 1 standard I/O port 2 value capture.*

### 4.40.1 Detailed Description

DM7820\_Register\_AdvInt\_0

## 4.41 DM7820 register incremental encoder 0 offsets

### Macros

- #define `DM7820_BAR2_INCENC0_ID` 0x0280  
*Incremental encoder 0 block identifier.*
- #define `DM7820_BAR2_INCENC0_INT` 0x0282  
*Incremental encoder 0 interrupt control/status.*
- #define `DM7820_BAR2_INCENC0_CLOCK` 0x0284  
*Incremental encoder 0 master clock.*
- #define `DM7820_BAR2_INCENC0_MODE` 0x0286  
*Incremental encoder 0 operating mode.*
- #define `DM7820_BAR2_INCENC0_VALUEA` 0x0288  
*Incremental encoder 0 channel A value.*
- #define `DM7820_BAR2_INCENC0_VALUEB` 0x028A  
*Incremental encoder 0 channel B value.*

### 4.41.1 Detailed Description

DM7820\_Register\_AdvInt\_1



## 4.42 DM7820 register incremental encoder 1 offsets

### Macros

- #define `DM7820_BAR2_INCENC1_ID` 0x02C0  
*Incremental encoder 1 block identifier.*
- #define `DM7820_BAR2_INCENC1_INT` 0x02C2  
*Incremental encoder 1 interrupt control/status.*
- #define `DM7820_BAR2_INCENC1_CLOCK` 0x02C4  
*Incremental encoder 1 master clock.*
- #define `DM7820_BAR2_INCENC1_MODE` 0x02C6  
*Incremental encoder 1 operating mode.*
- #define `DM7820_BAR2_INCENC1_VALUEA` 0x02C8  
*Incremental encoder 1 channel A value.*
- #define `DM7820_BAR2_INCENC1_VALUEB` 0x02CA  
*Incremental encoder 1 channel B value.*

### 4.42.1 Detailed Description

DM7820\_Register\_IncEnc\_0

## 4.43 DM7820 register pulse width modulator 0 offsets

### Macros

- #define [DM7820\\_BAR2\\_PWM0\\_ID](#) 0x0300  
*Pulse width modulator 0 block identifier.*
- #define [DM7820\\_BAR2\\_PWM0\\_MODE](#) 0x0302  
*Pulse width modulator 0 mode.*
- #define [DM7820\\_BAR2\\_PWM0\\_CLK](#) 0x0304  
*Pulse width modulator 0 period/width clocks.*
- #define [DM7820\\_BAR2\\_PWM0\\_PERIOD](#) 0x0308  
*Pulse width modulator 0 period.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHA](#) 0x0310  
*Pulse width modulator 0 output A width.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHB](#) 0x0314  
*Pulse width modulator 0 output B width.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHC](#) 0x0318  
*Pulse width modulator 0 output C width.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHD](#) 0x031C  
*Pulse width modulator 0 output D width.*

### 4.43.1 Detailed Description

DM7820\_Register\_IncEnc\_1

## 4.44 DM7820 register pulse width modulator 1 offsets

### Macros

- #define `DM7820_BAR2_PWM1_ID` 0x0340  
*Pulse width modulator 1 block identifier.*
- #define `DM7820_BAR2_PWM1_MODE` 0x0342  
*Pulse width modulator 1 mode.*
- #define `DM7820_BAR2_PWM1_CLK` 0x0344  
*Pulse width modulator 1 period/width clocks.*
- #define `DM7820_BAR2_PWM1_PERIOD` 0x0348  
*Pulse width modulator 1 period.*
- #define `DM7820_BAR2_PWM1_WIDTHA` 0x0350  
*Pulse width modulator 1 output A width.*
- #define `DM7820_BAR2_PWM1_WIDTHB` 0x0354  
*Pulse width modulator 1 output B width.*
- #define `DM7820_BAR2_PWM1_WIDTHC` 0x0358  
*Pulse width modulator 1 output C width.*
- #define `DM7820_BAR2_PWM1_WIDTHD` 0x035C  
*Pulse width modulator 1 output D width.*

### 4.44.1 Detailed Description

DM7820\_Register\_PWM\_0

## 4.45 DM7820 register 8254 timer/counter A offsets

### Macros

- #define [DM7820\\_BAR2\\_TCA\\_COUNTER\\_0](#) 0x1000  
*8254 timer/counter A timer 0 value*
- #define [DM7820\\_BAR2\\_TCA\\_COUNTER\\_1](#) 0x1004  
*8254 timer/counter A timer 1 value*
- #define [DM7820\\_BAR2\\_TCA\\_COUNTER\\_2](#) 0x1008  
*8254 timer/counter A timer 2 value*
- #define [DM7820\\_BAR2\\_TCA\\_CON\\_WORD](#) 0x100C  
*8254 timer/counter A control word*

### 4.45.1 Detailed Description

DM7820\_Register\_PWM\_1

## 4.46 DM7820 register 8254 timer/counter B offsets

### Macros

- #define [DM7820\\_BAR2\\_TCB\\_COUNTER\\_0](#) 0x1010  
*8254 timer/counter B timer 0 value*
- #define [DM7820\\_BAR2\\_TCB\\_COUNTER\\_1](#) 0x1014  
*8254 timer/counter B timer 1 value*
- #define [DM7820\\_BAR2\\_TCB\\_COUNTER\\_2](#) 0x1018  
*8254 timer/counter B timer 2 value*
- #define [DM7820\\_BAR2\\_TCB\\_CON\\_WORD](#) 0x101C  
*8254 timer/counter B control word*

### 4.46.1 Detailed Description

DM7820\_Register\_TmrCtr\_A

## 4.47 DM7820 register PCI region lengths

### Macros

- #define `DM7820_BAR0_LENGTH` 0x200  
*Length in bytes of BAR0 (memory-mapped PLX registers)*
- #define `DM7820_BAR1_LENGTH` 0x100  
*Length in bytes of BAR1 (I/O-mapped PLX registers)*
- #define `DM7820_BAR2_LENGTH` 0x2000  
*Length in bytes of BAR2 (memory-mapped FPGA registers)*

### 4.47.1 Detailed Description

DM7820\_Register\_BAR2\_Offsets

## 4.48 DM7820 register functional block identifiers

### Modules

- [DM7820 register functional block identifier values](#)
- [DM7820 register functional block identifier offsets](#)

### 4.48.1 Detailed Description

DM7820\_Register\_PCI\_Region\_Lengths

## 4.49 DM7820 register functional block identifier values

### Macros

- #define `DM7820_ID_TIMER_COUNTER` 0x1001  
*8254 timer/counter block identifier*
- #define `DM7820_ID_FIFO` 0x2011  
*FIFO block identifier.*
- #define `DM7820_ID_PROGRAMMABLE_CLOCK` 0x1000  
*Programmable clock block identifier.*
- #define `DM7820_ID_ADVANCED_INTERRUPT` 0x0001  
*Advanced interrupt block identifier.*
- #define `DM7820_ID_INCREMENTAL_ENCODER` 0x0002  
*Incremental encoder block identifier.*
- #define `DM7820_ID_PULSE_WIDTH_MODULATOR` 0x0003  
*Pulse width modulator block identifier.*
- #define `DM7820_ID_NONE` 0x0000  
*Empty block identifier.*

### 4.49.1 Detailed Description



## 4.50 DM7820 register functional block identifier offsets

### Macros

- #define `DM7820_FIRST_ID_OFFSET` 0x0080  
*Offset of first possible block identifier.*
- #define `DM7820_LAST_ID_OFFSET` 0x03C0  
*Offset of last possible block identifier.*

### 4.50.1 Detailed Description

DM7820\_Register\_Block\_ID\_Values

## 4.51 DM7820 register BAR2 macros

### Modules

- [DM7820 register BAR2 FPGA Version Register](#)
- [DM7820 register BAR2 Board Reset Register](#)

### 4.51.1 Detailed Description

DM7820\_Register\_Block\_IDs

## 4.52 DM7820 register BAR2 FPGA Version Register

### Macros

- #define `DM7820_FPGA_VERSION_TYPE_ID_MASK` 0xFF00  
*Bit mask to extract FPGA type identifier.*
- #define `DM7820_FPGA_VERSION_VERSION_MASK` 0x00FF  
*Bit mask to extract FPGA version identifier.*

### 4.52.1 Detailed Description

## 4.53 DM7820 register BAR2 Board Reset Register

### Macros

- #define [DM7820\\_BOARD\\_RESET\\_DO\\_RESET](#) 0xA5A5  
*Value to write to cause a board reset.*

### 4.53.1 Detailed Description

DM7820\_Register\_BAR2\_FPGA\_Version

## 4.54 DM7820 type definition header file

### Modules

- [DM7820 type enumerations](#)
- [DM7820 type definition structures](#)
- [DM7820 type definition typedefs](#)

### 4.54.1 Detailed Description

## 4.55 DM7820 type enumerations

### Modules

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

### 4.55.1 Detailed Description

## 4.56 DM7820 type PCI enumerations

### Typedefs

- typedef enum `dm7820_pci_region_num` `dm7820_pci_region_num_t`  
*Standard PCI region number type.*
- typedef enum `dm7820_pci_region_access_size` `dm7820_pci_region_access_size_t`  
*Standard PCI region access size type.*

### Enumerations

- enum `dm7820_pci_region_num` { `DM7820_PCI_REGION_PLX_MEM` = 0, `DM7820_PCI_REGION_PLX_IO`, `DM7820_PCI_REGION_FPGA_MEM` }  
*Standard PCI region number.*
- enum `dm7820_pci_region_access_size` { `DM7820_PCI_REGION_ACCESS_8` = 0, `DM7820_PCI_REGION_ACCESS_16`, `DM7820_PCI_REGION_ACCESS_32` }  
*Desired size in bits of access to standard PCI region.*

#### 4.56.1 Detailed Description

#### 4.56.2 Enumeration Type Documentation

##### 4.56.2.1 enum `dm7820_pci_region_access_size`

Desired size in bits of access to standard PCI region.

Enumerator:

**`DM7820_PCI_REGION_ACCESS_8`** 8-bit access  
**`DM7820_PCI_REGION_ACCESS_16`** 16-bit access  
**`DM7820_PCI_REGION_ACCESS_32`** 32-bit access

Definition at line 94 of file `dm7820_types.h`.

##### 4.56.2.2 enum `dm7820_pci_region_num`

Standard PCI region number.

Enumerator:

**`DM7820_PCI_REGION_PLX_MEM`** Memory-mapped PLX registers (BAR0)  
**`DM7820_PCI_REGION_PLX_IO`** I/O-mapped PLX registers (BAR1)  
**`DM7820_PCI_REGION_FPGA_MEM`** Memory-mapped FPGA registers (BAR2)

Definition at line 61 of file `dm7820_types.h`.

## 4.57 DM7820 type standard I/O enumerations

### Typedefs

- typedef enum [\\_DM7820\\_StdIO\\_Port](#) [DM7820\\_StdIO\\_Port](#)  
*Standard I/O port type.*
- typedef enum [\\_DM7820\\_StdIO\\_IO\\_Mode](#) [DM7820\\_StdIO\\_IO\\_Mode](#)  
*Standard I/O port mode type.*
- typedef enum [\\_DM7820\\_StdIO\\_Periph\\_Mode](#) [DM7820\\_StdIO\\_Periph\\_Mode](#)  
*Standard I/O port peripheral output mode type.*
- typedef enum [\\_DM7820\\_StdIO\\_Strobe](#) [DM7820\\_StdIO\\_Strobe](#)  
*Standard I/O port strobe signal type.*

### Enumerations

- enum [\\_DM7820\\_StdIO\\_Port](#) { [DM7820\\_STDIO\\_PORT\\_0](#) = 0, [DM7820\\_STDIO\\_PORT\\_1](#), [DM7820\\_STDIO\\_PORT\\_2](#) }  
*Standard I/O ports.*
- enum [\\_DM7820\\_StdIO\\_IO\\_Mode](#) { [DM7820\\_STDIO\\_MODE\\_INPUT](#) = 0, [DM7820\\_STDIO\\_MODE\\_OUTPUT](#), [DM7820\\_STDIO\\_MODE\\_PER\\_OUT](#) }  
*Standard I/O port modes.*
- enum [\\_DM7820\\_StdIO\\_Periph\\_Mode](#) { [DM7820\\_STDIO\\_PERIPH\\_PWM](#) = 0x0, [DM7820\\_STDIO\\_PERIPH\\_CLK\\_OTHER](#), [DM7820\\_STDIO\\_PERIPH\\_FIFO\\_0](#), [DM7820\\_STDIO\\_PERIPH\\_FIFO\\_1](#) }  
*Standard I/O port peripheral output modes.*
- enum [\\_DM7820\\_StdIO\\_Strobe](#) { [DM7820\\_STDIO\\_STROBE\\_1](#) = 0, [DM7820\\_STDIO\\_STROBE\\_2](#) }  
*Strobe signals.*

### 4.57.1 Detailed Description

DM7820\_Types\_PCI\_Enumerations

### 4.57.2 Enumeration Type Documentation

#### 4.57.2.1 enum [\\_DM7820\\_StdIO\\_IO\\_Mode](#)

Standard I/O port modes.

Enumerator:

**[DM7820\\_STDIO\\_MODE\\_INPUT](#)** Input  
**[DM7820\\_STDIO\\_MODE\\_OUTPUT](#)** Output  
**[DM7820\\_STDIO\\_MODE\\_PER\\_OUT](#)** Peripheral output

Definition at line 174 of file dm7820\_types.h.

#### 4.57.2.2 enum [\\_DM7820\\_StdIO\\_Periph\\_Mode](#)

Standard I/O port peripheral output modes.

Enumerator:

**[DM7820\\_STDIO\\_PERIPH\\_PWM](#)** Pulse Width Modulator (PWM) mode; valid for port 2 only



***DM7820\_STDIO\_PERIPH\_CLK\_OTHER*** Clock/other mode; valid for port 2 only

***DM7820\_STDIO\_PERIPH\_FIFO\_0*** FIFO 0 mode; valid for all ports

***DM7820\_STDIO\_PERIPH\_FIFO\_1*** FIFO 1 mode; valid for all ports

Definition at line 207 of file dm7820\_types.h.

#### 4.57.2.3 enum \_DM7820\_StdIO\_Port

Standard I/O ports.

Enumerator:

***DM7820\_STDIO\_PORT\_0*** Port 0

***DM7820\_STDIO\_PORT\_1*** Port 1

***DM7820\_STDIO\_PORT\_2*** Port 2

Definition at line 141 of file dm7820\_types.h.

#### 4.57.2.4 enum \_DM7820\_StdIO\_Strobe

Strobe signals.

Enumerator:

***DM7820\_STDIO\_STROBE\_1*** Strobe signal 1

***DM7820\_STDIO\_STROBE\_2*** Strobe signal 2

Definition at line 246 of file dm7820\_types.h.

## 4.58 DM7820 type timer/counter enumerations

### Typedefs

- typedef enum [\\_dm7820\\_tmrctr\\_timer](#) [dm7820\\_tmrctr\\_timer](#)  
*8254 timer/counter type*
- typedef enum [\\_dm7820\\_tmrctr\\_clock](#) [dm7820\\_tmrctr\\_clock](#)  
*8254 timer/counter clock selector type*
- typedef enum [\\_dm7820\\_tmrctr\\_gate](#) [dm7820\\_tmrctr\\_gate](#)  
*8254 timer/counter gate selector type*
- typedef enum [\\_dm7820\\_tmrctr\\_waveform](#) [dm7820\\_tmrctr\\_waveform](#)  
*8254 timer/counter waveform mode selector type*
- typedef enum [\\_dm7820\\_tmrctr\\_count\\_mode](#) [dm7820\\_tmrctr\\_count\\_mode](#)  
*8254 timer/counter count mode selector type*

### Enumerations

- enum [\\_dm7820\\_tmrctr\\_timer](#) {  
[DM7820\\_TMRCTR\\_TIMER\\_A\\_0](#) = 0, [DM7820\\_TMRCTR\\_TIMER\\_A\\_1](#), [DM7820\\_TMRCTR\\_TIMER\\_A\\_2](#), [DM7820\\_TMRCTR\\_TIMER\\_B\\_0](#),  
[DM7820\\_TMRCTR\\_TIMER\\_B\\_1](#), [DM7820\\_TMRCTR\\_TIMER\\_B\\_2](#) }  
*8254 timers/counters*
- enum [\\_dm7820\\_tmrctr\\_clock](#) {  
[DM7820\\_TMRCTR\\_CLOCK\\_5\\_MHZ](#) = 0, [DM7820\\_TMRCTR\\_CLOCK\\_RESERVED](#), [DM7820\\_TMRCTR\\_CLOCK\\_8254\\_A\\_0](#), [DM7820\\_TMRCTR\\_CLOCK\\_8254\\_A\\_1](#),  
[DM7820\\_TMRCTR\\_CLOCK\\_8254\\_A\\_2](#), [DM7820\\_TMRCTR\\_CLOCK\\_8254\\_B\\_0](#), [DM7820\\_TMRCTR\\_CLOCK\\_8254\\_B\\_1](#), [DM7820\\_TMRCTR\\_CLOCK\\_8254\\_B\\_2](#),  
[DM7820\\_TMRCTR\\_CLOCK\\_PROG\\_CLOCK\\_0](#), [DM7820\\_TMRCTR\\_CLOCK\\_PROG\\_CLOCK\\_1](#), [DM7820\\_TMRCTR\\_CLOCK\\_PROG\\_CLOCK\\_2](#), [DM7820\\_TMRCTR\\_CLOCK\\_PROG\\_CLOCK\\_3](#),  
[DM7820\\_TMRCTR\\_CLOCK\\_STROBE\\_1](#), [DM7820\\_TMRCTR\\_CLOCK\\_STROBE\\_2](#), [DM7820\\_TMRCTR\\_CLOCK\\_INV\\_STROBE\\_1](#), [DM7820\\_TMRCTR\\_CLOCK\\_INV\\_STROBE\\_2](#) }  
*8254 timer/counter clock selectors*
- enum [\\_dm7820\\_tmrctr\\_gate](#) {  
[DM7820\\_TMRCTR\\_GATE\\_LOGIC\\_0](#) = 0, [DM7820\\_TMRCTR\\_GATE\\_LOGIC\\_1](#), [DM7820\\_TMRCTR\\_GATE\\_8254\\_A\\_0](#), [DM7820\\_TMRCTR\\_GATE\\_8254\\_A\\_1](#),  
[DM7820\\_TMRCTR\\_GATE\\_8254\\_A\\_2](#), [DM7820\\_TMRCTR\\_GATE\\_8254\\_B\\_0](#), [DM7820\\_TMRCTR\\_GATE\\_8254\\_B\\_1](#), [DM7820\\_TMRCTR\\_GATE\\_8254\\_B\\_2](#),  
[DM7820\\_TMRCTR\\_GATE\\_PROG\\_CLOCK\\_0](#), [DM7820\\_TMRCTR\\_GATE\\_PROG\\_CLOCK\\_1](#), [DM7820\\_TMRCTR\\_GATE\\_PROG\\_CLOCK\\_2](#), [DM7820\\_TMRCTR\\_GATE\\_PROG\\_CLOCK\\_3](#),  
[DM7820\\_TMRCTR\\_GATE\\_STROBE\\_1](#), [DM7820\\_TMRCTR\\_GATE\\_STROBE\\_2](#), [DM7820\\_TMRCTR\\_GATE\\_INV\\_STROBE\\_1](#), [DM7820\\_TMRCTR\\_GATE\\_INV\\_STROBE\\_2](#),  
[DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_0](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_1](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_2](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_3](#),  
[DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_4](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_5](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_6](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_7](#),  
[DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_8](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_9](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_10](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_11](#),  
[DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_12](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_13](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_14](#), [DM7820\\_TMRCTR\\_GATE\\_PORT\\_2\\_BIT\\_15](#) }  
*8254 timer/counter gate selectors*

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

*8254 timer/counter waveform mode selectors*

- enum `_dm7820_tmrctr_count_mode` { `DM7820_TMRCTR_COUNT_MODE_BINARY` = 0, `DM7820_TMRCTR_COUNT_MODE_BCD` }

*8254 timer/counter count mode selectors*

### 4.58.1 Detailed Description

DM7820\_Types\_StdIO\_Enumerations

### 4.58.2 Enumeration Type Documentation

#### 4.58.2.1 enum `_dm7820_tmrctr_clock`

8254 timer/counter clock selectors

Enumerator:

**`DM7820_TMRCTR_CLOCK_5_MHZ`** 5 MHz clock  
**`DM7820_TMRCTR_CLOCK_RESERVED`** Reserved; do not use  
**`DM7820_TMRCTR_CLOCK_8254_A_0`** 8254 timer/counter A0  
**`DM7820_TMRCTR_CLOCK_8254_A_1`** 8254 timer/counter A1  
**`DM7820_TMRCTR_CLOCK_8254_A_2`** 8254 timer/counter A2  
**`DM7820_TMRCTR_CLOCK_8254_B_0`** 8254 timer/counter B0  
**`DM7820_TMRCTR_CLOCK_8254_B_1`** 8254 timer/counter B1  
**`DM7820_TMRCTR_CLOCK_8254_B_2`** 8254 timer/counter B2  
**`DM7820_TMRCTR_CLOCK_PROG_CLOCK_0`** Programmable clock 0  
**`DM7820_TMRCTR_CLOCK_PROG_CLOCK_1`** Programmable clock 1  
**`DM7820_TMRCTR_CLOCK_PROG_CLOCK_2`** Programmable clock 2  
**`DM7820_TMRCTR_CLOCK_PROG_CLOCK_3`** Programmable clock 3  
**`DM7820_TMRCTR_CLOCK_STROBE_1`** Strobe signal 1  
**`DM7820_TMRCTR_CLOCK_STROBE_2`** Strobe signal 2  
**`DM7820_TMRCTR_CLOCK_INV_STROBE_1`** Inverted strobe signal 1  
**`DM7820_TMRCTR_CLOCK_INV_STROBE_2`** Inverted strobe signal 2

Definition at line 337 of file `dm7820_types.h`.

#### 4.58.2.2 enum `_dm7820_tmrctr_count_mode`

8254 timer/counter count mode selectors

Enumerator:

**`DM7820_TMRCTR_COUNT_MODE_BINARY`** 16-bit binary mode  
**`DM7820_TMRCTR_COUNT_MODE_BCD`** Binary Coded Decimal (BCD) mode

Definition at line 706 of file `dm7820_types.h`.

## 4.58.2.3 enum\_dm7820\_tmrctr\_gate

8254 timer/counter gate selectors

Enumerator:

**DM7820\_TMRCTR\_GATE\_LOGIC\_0** Logic 0  
**DM7820\_TMRCTR\_GATE\_LOGIC\_1** Logic 1  
**DM7820\_TMRCTR\_GATE\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_TMRCTR\_GATE\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_TMRCTR\_GATE\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_TMRCTR\_GATE\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_TMRCTR\_GATE\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_TMRCTR\_GATE\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_TMRCTR\_GATE\_STROBE\_1** Strobe signal 1  
**DM7820\_TMRCTR\_GATE\_STROBE\_2** Strobe signal 2  
**DM7820\_TMRCTR\_GATE\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_TMRCTR\_GATE\_INV\_STROBE\_2** Inverted strobe signal 2  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_0** Digital I/O port 2 bit 0  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_1** Digital I/O port 2 bit 1  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_2** Digital I/O port 2 bit 2  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_3** Digital I/O port 2 bit 3  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_4** Digital I/O port 2 bit 4  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_5** Digital I/O port 2 bit 5  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_6** Digital I/O port 2 bit 6  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_7** Digital I/O port 2 bit 7  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_8** Digital I/O port 2 bit 8  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_9** Digital I/O port 2 bit 9  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_10** Digital I/O port 2 bit 10  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_11** Digital I/O port 2 bit 11  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_12** Digital I/O port 2 bit 12  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_13** Digital I/O port 2 bit 13  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_14** Digital I/O port 2 bit 14  
**DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_15** Digital I/O port 2 bit 15

Definition at line 448 of file dm7820\_types.h.

## 4.58.2.4 enum\_dm7820\_tmrctr\_timer

8254 timers/counters

Enumerator:

**DM7820\_TMRCTR\_TIMER\_A\_0** Timer 0 on first 8254 chip

**DM7820\_TMRCTR\_TIMER\_A\_1** Timer 1 on first 8254 chip  
**DM7820\_TMRCTR\_TIMER\_A\_2** Timer 2 on first 8254 chip  
**DM7820\_TMRCTR\_TIMER\_B\_0** Timer 0 on second 8254 chip  
**DM7820\_TMRCTR\_TIMER\_B\_1** Timer 1 on second 8254 chip  
**DM7820\_TMRCTR\_TIMER\_B\_2** Timer 2 on second 8254 chip

Definition at line 286 of file dm7820\_types.h.

#### 4.58.2.5 enum\_dm7820\_tmrctr\_waveform

8254 timer/counter waveform mode selectors

Enumerator:

**DM7820\_TMRCTR\_WAVEFORM\_EVENT\_CTR** Event counter  
**DM7820\_TMRCTR\_WAVEFORM\_PROG\_ONE\_SHOT** Programmable one shot  
**DM7820\_TMRCTR\_WAVEFORM\_RATE\_GENERATOR** Rate generator  
**DM7820\_TMRCTR\_WAVEFORM\_SQUARE\_WAVE** Square wave generator  
**DM7820\_TMRCTR\_WAVEFORM\_SOFTWARE\_STROBE** Software triggered strobe  
**DM7820\_TMRCTR\_WAVEFORM\_HARDWARE\_STROBE** Hardware triggered strobe

Definition at line 655 of file dm7820\_types.h.

## 4.59 DM7820 type programmable clock enumerations

### Typedefs

- typedef enum [\\_dm7820\\_prgclk\\_clock](#) dm7820\_prgclk\_clock  
*Programmable clock type.*
- typedef enum [\\_dm7820\\_prgclk\\_mode](#) dm7820\_prgclk\_mode  
*Programmable clock mode type.*
- typedef enum [\\_dm7820\\_prgclk\\_master\\_clock](#) dm7820\_prgclk\_master\_clock  
*Programmable clock master clock type.*
- typedef enum [\\_dm7820\\_prgclk\\_start\\_trigger](#) dm7820\_prgclk\_start\_trigger  
*Programmable clock start trigger type.*
- typedef enum [\\_dm7820\\_prgclk\\_stop\\_trigger](#) dm7820\_prgclk\_stop\_trigger  
*Programmable clock stop trigger type.*

### Enumerations

- enum [\\_dm7820\\_prgclk\\_clock](#) { DM7820\_PRGCLK\_CLOCK\_0 = 0, DM7820\_PRGCLK\_CLOCK\_1, DM7820\_PRGCLK\_CLOCK\_2, DM7820\_PRGCLK\_CLOCK\_3 }  
*Programmable clocks.*
- enum [\\_dm7820\\_prgclk\\_mode](#) { DM7820\_PRGCLK\_MODE\_DISABLED = 0, DM7820\_PRGCLK\_MODE\_CONTINUOUS, DM7820\_PRGCLK\_MODE\_RESERVED, DM7820\_PRGCLK\_MODE\_ONE\_SHOT }  
*Programmable clock modes.*
- enum [\\_dm7820\\_prgclk\\_master\\_clock](#) { DM7820\_PRGCLK\_MASTER\_25\_MHZ = 0, DM7820\_PRGCLK\_MASTER\_SAMPLE\_CLOCK, DM7820\_PRGCLK\_MASTER\_8254\_A\_0, DM7820\_PRGCLK\_MASTER\_8254\_A\_1, DM7820\_PRGCLK\_MASTER\_8254\_A\_2, DM7820\_PRGCLK\_MASTER\_8254\_B\_0, DM7820\_PRGCLK\_MASTER\_8254\_B\_1, DM7820\_PRGCLK\_MASTER\_8254\_B\_2, DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_0, DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_1, DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_2, DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_3, DM7820\_PRGCLK\_MASTER\_STROBE\_1, DM7820\_PRGCLK\_MASTER\_STROBE\_2, DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_1, DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_2 }  
*Programmable clock master clocks.*
- enum [\\_dm7820\\_prgclk\\_start\\_trigger](#) { DM7820\_PRGCLK\_START\_IMMEDIATE = 0, DM7820\_PRGCLK\_START\_RESERVED\_1, DM7820\_PRGCLK\_START\_8254\_A\_0, DM7820\_PRGCLK\_START\_8254\_A\_1, DM7820\_PRGCLK\_START\_8254\_A\_2, DM7820\_PRGCLK\_START\_8254\_B\_0, DM7820\_PRGCLK\_START\_8254\_B\_1, DM7820\_PRGCLK\_START\_8254\_B\_2, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3, DM7820\_PRGCLK\_START\_STROBE\_1, DM7820\_PRGCLK\_START\_STROBE\_2, DM7820\_PRGCLK\_START\_INV\_STROBE\_1, DM7820\_PRGCLK\_START\_INV\_STROBE\_2, DM7820\_PRGCLK\_START\_ADVANCED\_INT\_0, DM7820\_PRGCLK\_START\_ADVANCED\_INT\_1, DM7820\_PRGCLK\_START\_8254\_INT, DM7820\_PRGCLK\_START\_RESERVED\_2, DM7820\_PRGCLK\_START\_INC\_ENCODER\_0\_INT, DM7820\_PRGCLK\_START\_INC\_ENCODER\_1\_INT, DM7820\_PRGCLK\_START\_RESERVED\_3, DM7820\_PRGCLK\_START\_RESERVED\_4, DM7820\_PRGCLK\_START\_PWM\_0\_INT, DM7820\_PRGCLK\_START\_PWM\_1\_INT, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0\_INT, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1\_INT, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2\_INT, DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3\_INT, DM7820\_PRGCLK\_START\_FIFO\_0\_INT, DM7820\_PRGCLK\_START\_FIFO\_1\_INT }  
*Programmable clock start triggers.*

```

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

```

*Programmable clock stop triggers.*

#### 4.59.1 Detailed Description

DM7820\_Types\_TmrCtr\_Enumerations

#### 4.59.2 Enumeration Type Documentation

##### 4.59.2.1 enum \_dm7820\_prgclk\_clock

Programmable clocks.

Enumerator:

```

DM7820_PRGCLK_CLOCK_0 Programmable clock 0
DM7820_PRGCLK_CLOCK_1 Programmable clock 1
DM7820_PRGCLK_CLOCK_2 Programmable clock 2
DM7820_PRGCLK_CLOCK_3 Programmable clock 3

```

Definition at line 746 of file dm7820\_types.h.

##### 4.59.2.2 enum \_dm7820\_prgclk\_master\_clock

Programmable clock master clocks.

Enumerator:

```

DM7820_PRGCLK_MASTER_25_MHZ 25 MHz clock
DM7820_PRGCLK_MASTER_SAMPLE_CLOCK Reserved; do not use
DM7820_PRGCLK_MASTER_8254_A_0 8254 timer/counter A0
DM7820_PRGCLK_MASTER_8254_A_1 8254 timer/counter A1
DM7820_PRGCLK_MASTER_8254_A_2 8254 timer/counter A2
DM7820_PRGCLK_MASTER_8254_B_0 8254 timer/counter B0
DM7820_PRGCLK_MASTER_8254_B_1 8254 timer/counter B1
DM7820_PRGCLK_MASTER_8254_B_2 8254 timer/counter B2
DM7820_PRGCLK_MASTER_PROG_CLOCK_0 Programmable clock 0

```

**DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_PRGCLK\_MASTER\_STROBE\_1** Strobe signal 1  
**DM7820\_PRGCLK\_MASTER\_STROBE\_2** Strobe signal 2  
**DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_2** Inverted strobe signal 2

Definition at line 824 of file dm7820\_types.h.

#### 4.59.2.3 enum \_dm7820\_prgclk\_mode

Programmable clock modes.

Enumerator:

**DM7820\_PRGCLK\_MODE\_DISABLED** Disabled  
**DM7820\_PRGCLK\_MODE\_CONTINUOUS** Continuous mode  
**DM7820\_PRGCLK\_MODE\_RESERVED** Reserved; do not use  
**DM7820\_PRGCLK\_MODE\_ONE\_SHOT** One shot mode

Definition at line 785 of file dm7820\_types.h.

#### 4.59.2.4 enum \_dm7820\_prgclk\_start\_trigger

Programmable clock start triggers.

Enumerator:

**DM7820\_PRGCLK\_START\_IMMEDIATE** Start the clock immediately  
**DM7820\_PRGCLK\_START\_RESERVED\_1** Reserved; do not use  
**DM7820\_PRGCLK\_START\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_PRGCLK\_START\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_PRGCLK\_START\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_PRGCLK\_START\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_PRGCLK\_START\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_PRGCLK\_START\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_PRGCLK\_START\_STROBE\_1** Strobe signal 1  
**DM7820\_PRGCLK\_START\_STROBE\_2** Strobe signal 2  
**DM7820\_PRGCLK\_START\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_PRGCLK\_START\_INV\_STROBE\_2** Inverted strobe signal 2  
**DM7820\_PRGCLK\_START\_ADVANCED\_INT\_0** Advanced interrupt 0  
**DM7820\_PRGCLK\_START\_ADVANCED\_INT\_1** Advanced interrupt 1  
**DM7820\_PRGCLK\_START\_8254\_INT** 8254 timer/counter interrupt  
**DM7820\_PRGCLK\_START\_RESERVED\_2** Reserved; do not use



***DM7820\_PRGCLK\_START\_INC\_ENCODER\_0\_INT*** Incremental Encoder 0 interrupt  
***DM7820\_PRGCLK\_START\_INC\_ENCODER\_1\_INT*** Incremental Encoder 1 interrupt  
***DM7820\_PRGCLK\_START\_RESERVED\_3*** Reserved; do not use  
***DM7820\_PRGCLK\_START\_RESERVED\_4*** Reserved; do not use  
***DM7820\_PRGCLK\_START\_PWM\_0\_INT*** Pulse Width Modulator 0 interrupt  
***DM7820\_PRGCLK\_START\_PWM\_1\_INT*** Pulse Width Modulator 1 interrupt  
***DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0\_INT*** Programmable clock 0 interrupt  
***DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1\_INT*** Programmable clock 1 interrupt  
***DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2\_INT*** Programmable clock 2 interrupt  
***DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3\_INT*** Programmable clock 3 interrupt  
***DM7820\_PRGCLK\_START\_FIFO\_0\_INT*** FIFO 0 interrupt  
***DM7820\_PRGCLK\_START\_FIFO\_1\_INT*** FIFO 1 interrupt

Definition at line 935 of file dm7820\_types.h.

#### 4.59.2.5 enum\_dm7820\_prgclk\_stop\_trigger

Programmable clock stop triggers.

Enumerator:

***DM7820\_PRGCLK\_STOP\_NONE*** Do not stop the clock  
***DM7820\_PRGCLK\_STOP\_RESERVED\_1*** Reserved; do not use  
***DM7820\_PRGCLK\_STOP\_8254\_A\_0*** 8254 timer/counter A0  
***DM7820\_PRGCLK\_STOP\_8254\_A\_1*** 8254 timer/counter A1  
***DM7820\_PRGCLK\_STOP\_8254\_A\_2*** 8254 timer/counter A2  
***DM7820\_PRGCLK\_STOP\_8254\_B\_0*** 8254 timer/counter B0  
***DM7820\_PRGCLK\_STOP\_8254\_B\_1*** 8254 timer/counter B1  
***DM7820\_PRGCLK\_STOP\_8254\_B\_2*** 8254 timer/counter B2  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_0*** Programmable clock 0  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_1*** Programmable clock 1  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_2*** Programmable clock 2  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_3*** Programmable clock 3  
***DM7820\_PRGCLK\_STOP\_STROBE\_1*** Strobe signal 1  
***DM7820\_PRGCLK\_STOP\_STROBE\_2*** Strobe signal 2  
***DM7820\_PRGCLK\_STOP\_INV\_STROBE\_1*** Inverted strobe signal 1  
***DM7820\_PRGCLK\_STOP\_INV\_STROBE\_2*** Inverted strobe signal 2  
***DM7820\_PRGCLK\_STOP\_ADVANCED\_INT\_0*** Advanced interrupt 0  
***DM7820\_PRGCLK\_STOP\_ADVANCED\_INT\_1*** Advanced interrupt 1  
***DM7820\_PRGCLK\_STOP\_8254\_INT*** 8254 timer/counter interrupt  
***DM7820\_PRGCLK\_STOP\_RESERVED\_2*** Reserved; do not use  
***DM7820\_PRGCLK\_STOP\_INC\_ENCODER\_0\_INT*** Incremental Encoder 0 interrupt  
***DM7820\_PRGCLK\_STOP\_INC\_ENCODER\_1\_INT*** Incremental Encoder 1 interrupt  
***DM7820\_PRGCLK\_STOP\_RESERVED\_3*** Reserved; do not use  
***DM7820\_PRGCLK\_STOP\_RESERVED\_4*** Reserved; do not use  
***DM7820\_PRGCLK\_STOP\_PWM\_0\_INT*** Pulse Width Modulator 0 interrupt  
***DM7820\_PRGCLK\_STOP\_PWM\_1\_INT*** Pulse Width Modulator 1 interrupt

***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_0\_INT*** Programmable clock 0 interrupt  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_1\_INT*** Programmable clock 1 interrupt  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_2\_INT*** Programmable clock 2 interrupt  
***DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_3\_INT*** Programmable clock 3 interrupt  
***DM7820\_PRGCLK\_STOP\_FIFO\_0\_INT*** FIFO 0 interrupt  
***DM7820\_PRGCLK\_STOP\_FIFO\_1\_INT*** FIFO 1 interrupt

Definition at line 1142 of file dm7820\_types.h.

## 4.60 DM7820 type pulse width modulator enumerations

### Typedefs

- typedef enum `_dm7820_pwm_modulator` `dm7820_pwm_modulator`  
*Pulse width modulator type.*
- typedef enum `_dm7820_pwm_period_master_clock` `dm7820_pwm_period_master_clock`  
*Pulse width modulator period master clock type.*
- typedef enum `_dm7820_pwm_output` `dm7820_pwm_output`  
*Pulse width modulator output type.*
- typedef enum `_dm7820_pwm_width_master_clock` `dm7820_pwm_width_master_clock`  
*Pulse width modulator width master clock type.*

### Enumerations

- enum `_dm7820_pwm_modulator`  
*Pulse width modulators.*
- enum `_dm7820_pwm_period_master_clock` {  
`DM7820_PWM_PERIOD_MASTER_25_MHZ = 0, DM7820_PWM_PERIOD_MASTER_RESERVED, D-`  
`M7820_PWM_PERIOD_MASTER_8254_A_0, DM7820_PWM_PERIOD_MASTER_8254_A_1,`  
`DM7820_PWM_PERIOD_MASTER_8254_A_2, DM7820_PWM_PERIOD_MASTER_8254_B_0, DM7820_`  
`PWM_PERIOD_MASTER_8254_B_1, DM7820_PWM_PERIOD_MASTER_8254_B_2,`  
`DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0, DM7820_PWM_PERIOD_MASTER_PROG_CLO-`  
`CK_1, DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_2, DM7820_PWM_PERIOD_MASTER_PRO-`  
`G_CLOCK_3,`  
`DM7820_PWM_PERIOD_MASTER_STROBE_1, DM7820_PWM_PERIOD_MASTER_STROBE_2, D-`  
`M7820_PWM_PERIOD_MASTER_INV_STROBE_1, DM7820_PWM_PERIOD_MASTER_INV_STROBE_2`  
 }  
*Pulse width modulator period master clocks.*
- enum `_dm7820_pwm_output` { `DM7820_PWM_OUTPUT_A = 0, DM7820_PWM_OUTPUT_B, DM7820_P-`  
`WM_OUTPUT_C, DM7820_PWM_OUTPUT_D` }  
*Pulse width modulator outputs.*
- enum `_dm7820_pwm_width_master_clock` {  
`DM7820_PWM_WIDTH_MASTER_25_MHZ = 0, DM7820_PWM_WIDTH_MASTER_RESERVED, DM7820-`  
`_PWM_WIDTH_MASTER_8254_A_0, DM7820_PWM_WIDTH_MASTER_8254_A_1,`  
`DM7820_PWM_WIDTH_MASTER_8254_A_2, DM7820_PWM_WIDTH_MASTER_8254_B_0, DM7820_P-`  
`WM_WIDTH_MASTER_8254_B_1, DM7820_PWM_WIDTH_MASTER_8254_B_2,`  
`DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0, DM7820_PWM_WIDTH_MASTER_PROG_CLO-`  
`CK_1, DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_2, DM7820_PWM_WIDTH_MASTER_PROG_CL-`  
`OCK_3,`  
`DM7820_PWM_WIDTH_MASTER_STROBE_1, DM7820_PWM_WIDTH_MASTER_STROBE_2, DM7820-`  
`_PWM_WIDTH_MASTER_INV_STROBE_1, DM7820_PWM_WIDTH_MASTER_INV_STROBE_2` }  
*Pulse width modulator width master clocks.*

#### 4.60.1 Detailed Description

DM7820\_Types\_PrgClk\_Enumerations

## 4.60.2 Enumeration Type Documentation

### 4.60.2.1 enum\_dm7820\_pwm\_output

Pulse width modulator outputs.

Enumerator:

**DM7820\_PWM\_OUTPUT\_A** PWM output A  
**DM7820\_PWM\_OUTPUT\_B** PWM output B  
**DM7820\_PWM\_OUTPUT\_C** PWM output C  
**DM7820\_PWM\_OUTPUT\_D** PWM output D

Definition at line 1501 of file dm7820\_types.h.

### 4.60.2.2 enum\_dm7820\_pwm\_period\_master\_clock

Pulse width modulator period master clocks.

Enumerator:

**DM7820\_PWM\_PERIOD\_MASTER\_25\_MHZ** 25 MHz clock  
**DM7820\_PWM\_PERIOD\_MASTER\_RESERVED** Reserved; do not use  
**DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_PWM\_PERIOD\_MASTER\_STROBE\_1** Strobe signal 1  
**DM7820\_PWM\_PERIOD\_MASTER\_STROBE\_2** Strobe signal 2  
**DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_2** Inverted strobe signal 2

Definition at line 1389 of file dm7820\_types.h.

### 4.60.2.3 enum\_dm7820\_pwm\_width\_master\_clock

Pulse width modulator width master clocks.

Enumerator:

**DM7820\_PWM\_WIDTH\_MASTER\_25\_MHZ** 25 MHz clock  
**DM7820\_PWM\_WIDTH\_MASTER\_RESERVED** Reserved; do not use  
**DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_2** 8254 timer/counter A2

***DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_0*** 8254 timer/counter B0  
***DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_1*** 8254 timer/counter B1  
***DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_2*** 8254 timer/counter B2  
***DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_0*** Programmable clock 0  
***DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_1*** Programmable clock 1  
***DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_2*** Programmable clock 2  
***DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_3*** Programmable clock 3  
***DM7820\_PWM\_WIDTH\_MASTER\_STROBE\_1*** Strobe signal 1  
***DM7820\_PWM\_WIDTH\_MASTER\_STROBE\_2*** Strobe signal 2  
***DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_1*** Inverted strobe signal 1  
***DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_2*** Inverted strobe signal 2

Definition at line 1540 of file dm7820\_types.h.

## 4.61 DM7820 type incremental encoder enumerations

### Typedefs

- typedef enum [\\_dm7820\\_incenc\\_encoder](#) [dm7820\\_incenc\\_encoder](#)  
*Incremental encoder type.*
- typedef enum [\\_dm7820\\_incenc\\_master\\_clock](#) [dm7820\\_incenc\\_master\\_clock](#)  
*Incremental encoder master clock type.*
- typedef enum [\\_dm7820\\_incenc\\_input\\_mode](#) [dm7820\\_incenc\\_input\\_mode](#)  
*Incremental encoder input mode type.*
- typedef enum [\\_dm7820\\_incenc\\_channel\\_mode](#) [dm7820\\_incenc\\_channel\\_mode](#)  
*Incremental encoder channel mode type.*
- typedef enum [\\_dm7820\\_incenc\\_phase\\_transition](#) [dm7820\\_incenc\\_phase\\_transition](#)  
*Incremental encoder phase filter transition type.*
- typedef enum [\\_dm7820\\_incenc\\_channel](#) [dm7820\\_incenc\\_channel](#)  
*Incremental encoder channel type.*
- typedef enum [\\_dm7820\\_incenc\\_status\\_condition](#) [dm7820\\_incenc\\_status\\_condition](#)  
*Incremental encoder status condition type.*

### Enumerations

- enum [\\_dm7820\\_incenc\\_encoder](#) { [DM7820\\_INCENC\\_ENCODER\\_0](#) = 0, [DM7820\\_INCENC\\_ENCODER\\_1](#) }  
*Incremental encoders.*
- enum [\\_dm7820\\_incenc\\_master\\_clock](#) { [DM7820\\_INCENC\\_MASTER\\_25\\_MHZ](#) = 0, [DM7820\\_INCENC\\_MASTER\\_RESERVED](#), [DM7820\\_INCENC\\_MASTER\\_8254\\_A\\_0](#), [DM7820\\_INCENC\\_MASTER\\_8254\\_A\\_1](#), [DM7820\\_INCENC\\_MASTER\\_8254\\_A\\_2](#), [DM7820\\_INCENC\\_MASTER\\_8254\\_B\\_0](#), [DM7820\\_INCENC\\_MASTER\\_8254\\_B\\_1](#), [DM7820\\_INCENC\\_MASTER\\_8254\\_B\\_2](#), [DM7820\\_INCENC\\_MASTER\\_PROG\\_CLOCK\\_0](#), [DM7820\\_INCENC\\_MASTER\\_PROG\\_CLOCK\\_1](#), [DM7820\\_INCENC\\_MASTER\\_PROG\\_CLOCK\\_2](#), [DM7820\\_INCENC\\_MASTER\\_PROG\\_CLOCK\\_3](#), [DM7820\\_INCENC\\_MASTER\\_STROBE\\_1](#), [DM7820\\_INCENC\\_MASTER\\_STROBE\\_2](#), [DM7820\\_INCENC\\_MASTER\\_INV\\_STROBE\\_1](#), [DM7820\\_INCENC\\_MASTER\\_INV\\_STROBE\\_2](#) }  
*Incremental encoder master clocks.*
- enum [\\_dm7820\\_incenc\\_input\\_mode](#) { [DM7820\\_INCENC\\_INPUT\\_SINGLE\\_ENDED](#) = 0, [DM7820\\_INCENC\\_INPUT\\_DIFFERENTIAL](#) }  
*Incremental encoder input modes.*
- enum [\\_dm7820\\_incenc\\_channel\\_mode](#) { [DM7820\\_INCENC\\_CHANNEL\\_INDEPENDENT](#) = 0, [DM7820\\_INCENC\\_CHANNEL\\_JOINED](#) }  
*Incremental encoder channel modes.*
- enum [\\_dm7820\\_incenc\\_phase\\_transition](#) { [DM7820\\_INCENC\\_PHASE\\_BA\\_00\\_TO\\_01\\_UP](#) = 0, [DM7820\\_INCENC\\_PHASE\\_BA\\_01\\_TO\\_11\\_UP](#), [DM7820\\_INCENC\\_PHASE\\_BA\\_11\\_TO\\_10\\_UP](#), [DM7820\\_INCENC\\_PHASE\\_BA\\_10\\_TO\\_00\\_UP](#), [DM7820\\_INCENC\\_PHASE\\_BA\\_01\\_TO\\_00\\_DOWN](#), [DM7820\\_INCENC\\_PHASE\\_BA\\_11\\_TO\\_01\\_DOWN](#), [DM7820\\_INCENC\\_PHASE\\_BA\\_10\\_TO\\_11\\_DOWN](#), [DM7820\\_INCENC\\_PHASE\\_BA\\_00\\_TO\\_10\\_DOWN](#) }  
*Incremental encoder phase filter transitions.*
- enum [\\_dm7820\\_incenc\\_channel](#) { [DM7820\\_INCENC\\_CHANNEL\\_A](#) = 0, [DM7820\\_INCENC\\_CHANNEL\\_B](#) }

*Incremental encoder channels.*

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

*Incremental encoder status conditions.*

#### 4.61.1 Detailed Description

DM7820\_Types\_PWM\_Enumerations

#### 4.61.2 Enumeration Type Documentation

##### 4.61.2.1 enum `_dm7820_incenc_channel`

Incremental encoder channels.

Enumerator:

**`DM7820_INCENC_CHANNEL_A`** Channel A

**`DM7820_INCENC_CHANNEL_B`** Channel B

Definition at line 1921 of file `dm7820_types.h`.

##### 4.61.2.2 enum `_dm7820_incenc_channel_mode`

Incremental encoder channel modes.

Enumerator:

**`DM7820_INCENC_CHANNEL_INDEPENDENT`** Independent 16-bit channels

**`DM7820_INCENC_CHANNEL_JOINED`** Channels joined into single 32-bit channel

Definition at line 1830 of file `dm7820_types.h`.

##### 4.61.2.3 enum `_dm7820_incenc_encoder`

Incremental encoders.

Enumerator:

**`DM7820_INCENC_ENCODER_0`** Incremental encoder 0

**`DM7820_INCENC_ENCODER_1`** Incremental encoder 1

Definition at line 1665 of file `dm7820_types.h`.

##### 4.61.2.4 enum `_dm7820_incenc_input_mode`

Incremental encoder input modes.

Enumerator:

**`DM7820_INCENC_INPUT_SINGLE_ENDED`** Single ended

**`DM7820_INCENC_INPUT_DIFFERENTIAL`** Pseudo differential

Definition at line 1803 of file `dm7820_types.h`.

## 4.61.2.5 enum\_dm7820\_incenc\_master\_clock

Incremental encoder master clocks.

Enumerator:

**DM7820\_INCENC\_MASTER\_25\_MHZ** 25 MHz clock  
**DM7820\_INCENC\_MASTER\_RESERVED** Reserved; do not use  
**DM7820\_INCENC\_MASTER\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_INCENC\_MASTER\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_INCENC\_MASTER\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_INCENC\_MASTER\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_INCENC\_MASTER\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_INCENC\_MASTER\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_INCENC\_MASTER\_STROBE\_1** Strobe signal 1  
**DM7820\_INCENC\_MASTER\_STROBE\_2** Strobe signal 2  
**DM7820\_INCENC\_MASTER\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_INCENC\_MASTER\_INV\_STROBE\_2** Inverted strobe signal 2

Definition at line 1692 of file dm7820\_types.h.

## 4.61.2.6 enum\_dm7820\_incenc\_phase\_transition

Incremental encoder phase filter transitions.

Enumerator:

**DM7820\_INCENC\_PHASE\_BA\_00\_TO\_01\_UP** Inputs B/A transition from 0/0 to 0/1 when counting up  
**DM7820\_INCENC\_PHASE\_BA\_01\_TO\_11\_UP** Inputs B/A transition from 0/1 to 1/1 when counting up  
**DM7820\_INCENC\_PHASE\_BA\_11\_TO\_10\_UP** Inputs B/A transition from 1/1 to 1/0 when counting up  
**DM7820\_INCENC\_PHASE\_BA\_10\_TO\_00\_UP** Inputs B/A transition from 1/0 to 0/0 when counting up  
**DM7820\_INCENC\_PHASE\_BA\_01\_TO\_00\_DOWN** Inputs B/A transition from 0/1 to 0/0 when counting down  
  
**DM7820\_INCENC\_PHASE\_BA\_11\_TO\_01\_DOWN** Inputs B/A transition from 1/1 to 0/1 when counting down  
  
**DM7820\_INCENC\_PHASE\_BA\_10\_TO\_11\_DOWN** Inputs B/A transition from 1/0 to 1/1 when counting down  
  
**DM7820\_INCENC\_PHASE\_BA\_00\_TO\_10\_DOWN** Inputs B/A transition from 0/0 to 1/0 when counting down

Definition at line 1857 of file dm7820\_types.h.



## 4.61.2.7 enum\_dm7820\_incenc\_status\_condition

Incremental encoder status conditions.

Enumerator:

**DM7820\_INCENC\_STATUS\_CHANNEL\_A\_POSITIVE\_ROLLOVER** Channel A positive rollover  
**DM7820\_INCENC\_STATUS\_CHANNEL\_A\_NEGATIVE\_ROLLOVER** Channel A negative rollover  
**DM7820\_INCENC\_STATUS\_CHANNEL\_B\_POSITIVE\_ROLLOVER** Channel B positive rollover  
**DM7820\_INCENC\_STATUS\_CHANNEL\_B\_NEGATIVE\_ROLLOVER** Channel B negative rollover

Definition at line 1948 of file dm7820\_types.h.

## 4.62 DM7820 type FIFO enumerations

### Typedefs

- typedef enum [\\_dm7820\\_fifo\\_queue](#) [dm7820\\_fifo\\_queue](#)  
*FIFO type.*
- typedef enum [\\_dm7820\\_fifo\\_input\\_clock](#) [dm7820\\_fifo\\_input\\_clock](#)  
*FIFO input clock type.*
- typedef enum [\\_dm7820\\_fifo\\_output\\_clock](#) [dm7820\\_fifo\\_output\\_clock](#)  
*FIFO output clock type.*
- typedef enum [\\_dm7820\\_fifo\\_dma\\_request](#) [dm7820\\_fifo\\_dma\\_request](#)  
*FIFO DMA request source type.*
- typedef enum [\\_dm7820\\_fifo\\_data\\_input](#) [dm7820\\_fifo\\_data\\_input](#)  
*FIFO data input type.*
- typedef enum [\\_dm7820\\_fifo\\_status\\_condition](#) [dm7820\\_fifo\\_status\\_condition](#)  
*FIFO status condition type.*

### Enumerations

- enum [\\_dm7820\\_fifo\\_queue](#) { [DM7820\\_FIFO\\_QUEUE\\_0](#) = 0, [DM7820\\_FIFO\\_QUEUE\\_1](#) }  
*FIFOs.*
- enum [\\_dm7820\\_fifo\\_input\\_clock](#) {  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_25\\_MHZ](#) = 0, [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_RESERVED\\_1](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_A\\_0](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_A\\_1](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_A\\_2](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_B\\_0](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_B\\_1](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_B\\_2](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_0](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_1](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_2](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_3](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_STROBE\\_1](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_STROBE\\_2](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_INV\\_STROBE\\_1](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_INV\\_STROBE\\_2](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_ADVANCED\\_INT\\_0](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_ADVANCED\\_INT\\_1](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_8254\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_RESERVED\\_2](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_INC\\_ENCODER\\_0\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_INC\\_ENCODER\\_1\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_RESERVED\\_3](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_RESERVED\\_4](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PWM\\_0\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PWM\\_1\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_0\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_1\\_INT](#),  
[DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_2\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_3\\_INT](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PCI\\_READ](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PCI\\_WRITE](#) }  
*FIFO input clocks.*
- enum [\\_dm7820\\_fifo\\_output\\_clock](#) {  
[DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_25\\_MHZ](#) = 0, [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_RESERVED\\_1](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_8254\\_A\\_0](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_8254\\_A\\_1](#),  
[DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_8254\\_A\\_2](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_8254\\_B\\_0](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_8254\\_B\\_1](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_8254\\_B\\_2](#),  
[DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_PROG\\_CLOCK\\_0](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_PROG\\_CLOCK\\_1](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_PROG\\_CLOCK\\_2](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_PROG\\_CLOCK\\_3](#),  
[DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_STROBE\\_1](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_STROBE\\_2](#), [DM7820-](#)

```
_FIFO_OUTPUT_CLOCK_INV_STROBE_1, DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_2,
DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_0, DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_I-
NT_1, DM7820_FIFO_OUTPUT_CLOCK_8254_INT, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2,
DM7280_FIFO_OUTPUT_CLOCK_INC_ENCODER_0_INT, DM7280_FIFO_OUTPUT_CLOCK_INC_ENC-
ODER_1_INT, DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3, DM7820_FIFO_OUTPUT_CLOCK_RE-
SERVED_4,
DM7820_FIFO_OUTPUT_CLOCK_PWM_0_INT, DM7820_FIFO_OUTPUT_CLOCK_PWM_1_INT, D-
M7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CL-
OCK_1_INT,
DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2_INT, DM7820_FIFO_OUTPUT_CLOCK_PROG_CL-
OCK_3_INT, DM7820_FIFO_OUTPUT_CLOCK_PCI_READ, DM7820_FIFO_OUTPUT_CLOCK_PCI_WRI-
TE }
```

*FIFO output clocks.*

- enum `_dm7820_fifo_dma_request` { `DM7820_FIFO_DMA_REQUEST_READ` = 0, `DM7820_FIFO_DMA_R-REQUEST_NOT_EMPTY`, `DM7820_FIFO_DMA_REQUEST_WRITE`, `DM7820_FIFO_DMA_REQUEST_NO-T_FULL` }

*FIFO DMA request sources.*

- enum `_dm7820_fifo_data_input` { `DM7820_FIFO_0_DATA_INPUT_PCI_DATA` = 0, `DM7820_FIFO_0_DATA_INPUT_PORT_0`, `DM7820_FIF-O_0_DATA_INPUT_PORT_2`, `DM7820_FIFO_0_DATA_INPUT_FIFO_0_OUTPUT`, `DM7820_FIFO_1_DATA_INPUT_PCI_DATA`, `DM7820_FIFO_1_DATA_INPUT_PORT_1`, `DM7820_FIFO_-1_DATA_INPUT_INC_ENCODER_1_A`, `DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_B` }

*FIFO data inputs.*

- enum `_dm7820_fifo_status_condition` { `DM7820_FIFO_STATUS_READ_REQUEST` = 0, `DM7820_FIFO_STATUS_WRITE_REQUEST`, `DM7820_-FIFO_STATUS_FULL`, `DM7820_FIFO_STATUS_EMPTY`, `DM7820_FIFO_STATUS_OVERFLOW`, `DM7820_FIFO_STATUS_UNDERFLOW` }

*FIFO status conditions.*

### 4.62.1 Detailed Description

DM7820\_Types\_IncEnc\_Enumerations

### 4.62.2 Enumeration Type Documentation

#### 4.62.2.1 enum `_dm7820_fifo_data_input`

FIFO data inputs.

Enumerator:

**`DM7820_FIFO_0_DATA_INPUT_PCI_DATA`** FIFO 0 data input is PCI data

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

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

**`DM7820_FIFO_0_DATA_INPUT_FIFO_0_OUTPUT`** FIFO 0 data input is FIFO 0 output

**`DM7820_FIFO_1_DATA_INPUT_PCI_DATA`** FIFO 1 data input is PCI data

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

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

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

Definition at line 2481 of file `dm7820_types.h`.

## 4.62.2.2 enum\_dm7820\_fifo\_dma\_request

FIFO DMA request sources.

Enumerator:

**DM7820\_FIFO\_DMA\_REQUEST\_READ** Read request  
**DM7820\_FIFO\_DMA\_REQUEST\_NOT\_EMPTY** Not empty  
**DM7820\_FIFO\_DMA\_REQUEST\_WRITE** Write request  
**DM7820\_FIFO\_DMA\_REQUEST\_NOT\_FULL** Not full

Definition at line 2442 of file dm7820\_types.h.

## 4.62.2.3 enum\_dm7820\_fifo\_input\_clock

FIFO input clocks.

Enumerator:

**DM7820\_FIFO\_INPUT\_CLOCK\_25\_MHZ** 25 MHz clock  
**DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_1** Reserved; do not use  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_FIFO\_INPUT\_CLOCK\_STROBE\_1** Strobe signal 1  
**DM7820\_FIFO\_INPUT\_CLOCK\_STROBE\_2** Strobe signal 2  
**DM7820\_FIFO\_INPUT\_CLOCK\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_FIFO\_INPUT\_CLOCK\_INV\_STROBE\_2** Inverted strobe signal 2  
**DM7820\_FIFO\_INPUT\_CLOCK\_ADVANCED\_INT\_0** Advanced interrupt 0  
**DM7820\_FIFO\_INPUT\_CLOCK\_ADVANCED\_INT\_1** Advanced interrupt 1  
**DM7820\_FIFO\_INPUT\_CLOCK\_8254\_INT** 8254 timer/counter interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_2** Reserved; do not use  
**DM7820\_FIFO\_INPUT\_CLOCK\_INC\_ENCODER\_0\_INT** Incremental encoder 0 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_INC\_ENCODER\_1\_INT** Incremental encoder 1 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_3** Reserved; do not use  
**DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_4** Reserved; do not use  
**DM7820\_FIFO\_INPUT\_CLOCK\_PWM\_0\_INT** PWM 0 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_PWM\_1\_INT** PWM 1 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0\_INT** Programmable clock 0 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_1\_INT** Programmable clock 1 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_2\_INT** Programmable clock 2 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_3\_INT** Programmable clock 3 interrupt  
**DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_READ** PCI read from FIFO  
**DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_WRITE** PCI write to FIFO

Definition at line 2028 of file dm7820\_types.h.

## 4.62.2.4 enum\_dm7820\_fifo\_output\_clock

FIFO output clocks.

Enumerator:

**DM7820\_FIFO\_OUTPUT\_CLOCK\_25\_MHZ** 25 MHz clock  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_1** Reserved; do not use  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_STROBE\_1** Strobe signal 1  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_STROBE\_2** Strobe signal 2  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_INV\_STROBE\_2** Inverted strobe signal 2  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_ADVANCED\_INT\_0** Advanced interrupt 0  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_ADVANCED\_INT\_1** Advanced interrupt 1  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_INT** 8254 timer/counter interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_2** Reserved; do not use  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_INC\_ENCODER\_0\_INT** Incremental encoder 0 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_INC\_ENCODER\_1\_INT** Incremental encoder 1 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_3** Reserved; do not use  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_4** Reserved; do not use  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PWM\_0\_INT** PWM 0 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PWM\_1\_INT** PWM 1 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0\_INT** Programmable clock 0 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_1\_INT** Programmable clock 1 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_2\_INT** Programmable clock 2 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_3\_INT** Programmable clock 3 interrupt  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_READ** PCI read from FIFO  
**DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_WRITE** PCI write to FIFO

Definition at line 2235 of file dm7820\_types.h.

## 4.62.2.5 enum\_dm7820\_fifo\_queue

FIFOs.

Enumerator:

**DM7820\_FIFO\_QUEUE\_0** FIFO 0  
**DM7820\_FIFO\_QUEUE\_1** FIFO 1

Definition at line 2001 of file dm7820\_types.h.

#### 4.62.2.6 enum\_dm7820\_fifo\_status\_condition

FIFO status conditions.

Enumerator:

***DM7820\_FIFO\_STATUS\_READ\_REQUEST*** FIFO read request  
***DM7820\_FIFO\_STATUS\_WRITE\_REQUEST*** FIFO read request  
***DM7820\_FIFO\_STATUS\_FULL*** FIFO full  
***DM7820\_FIFO\_STATUS\_EMPTY*** FIFO empty  
***DM7820\_FIFO\_STATUS\_OVERFLOW*** FIFO overflow  
***DM7820\_FIFO\_STATUS\_UNDERFLOW*** FIFO underflow

Definition at line 2544 of file dm7820\_types.h.

## 4.63 DM7820 type advanced interrupt enumerations

### Typedefs

- typedef enum `_dm7820_advint_interrupt dm7820_advint_interrupt`  
*Advanced interrupt type.*
- typedef enum `_dm7820_advint_mode dm7820_advint_mode`  
*Advanced interrupt mode type.*
- typedef enum `_dm7820_advint_master_clock dm7820_advint_master_clock`  
*Advanced interrupt master clock type.*

### Enumerations

- enum `_dm7820_advint_interrupt` { `DM7820_ADVINT_INTERRUPT_0` = 0, `DM7820_ADVINT_INTERRUPT_1` }  
*Advanced interrupts.*
- enum `_dm7820_advint_mode` { `DM7820_ADVINT_MODE_DISABLED` = 0, `DM7820_ADVINT_MODE_STROBE`, `DM7820_ADVINT_MODE_MATCH`, `DM7820_ADVINT_MODE_EVENT` }  
*Advanced interrupt modes.*
- enum `_dm7820_advint_master_clock` {  
`DM7820_ADVINT_MASTER_25_MHZ` = 0, `DM7820_ADVINT_MASTER_RESERVED`, `DM7820_ADVINT_MASTER_8254_A_0`, `DM7820_ADVINT_MASTER_8254_A_1`,  
`DM7820_ADVINT_MASTER_8254_A_2`, `DM7820_ADVINT_MASTER_8254_B_0`, `DM7820_ADVINT_MASTER_8254_B_1`, `DM7820_ADVINT_MASTER_8254_B_2`,  
`DM7820_ADVINT_MASTER_PROG_CLOCK_0`, `DM7820_ADVINT_MASTER_PROG_CLOCK_1`, `DM7820_ADVINT_MASTER_PROG_CLOCK_2`, `DM7820_ADVINT_MASTER_PROG_CLOCK_3`,  
`DM7820_ADVINT_MASTER_STROBE_1`, `DM7820_ADVINT_MASTER_STROBE_2`, `DM7820_ADVINT_MASTER_INV_STROBE_1`, `DM7820_ADVINT_MASTER_INV_STROBE_2` }  
*Advanced interrupt master clocks.*

#### 4.63.1 Detailed Description

DM7820\_Types\_FIFO\_Enumerations

#### 4.63.2 Enumeration Type Documentation

##### 4.63.2.1 enum `_dm7820_advint_interrupt`

Advanced interrupts.

Enumerator:

**`DM7820_ADVINT_INTERRUPT_0`** Advanced interrupt 0

**`DM7820_ADVINT_INTERRUPT_1`** Advanced interrupt 1

Definition at line 2608 of file `dm7820_types.h`.

#### 4.63.2.2 enum\_dm7820\_advint\_master\_clock

Advanced interrupt master clocks.

Enumerator:

**DM7820\_ADVINT\_MASTER\_25\_MHZ** 25 MHz clock  
**DM7820\_ADVINT\_MASTER\_RESERVED** Reserved; do not use  
**DM7820\_ADVINT\_MASTER\_8254\_A\_0** 8254 timer/counter A0  
**DM7820\_ADVINT\_MASTER\_8254\_A\_1** 8254 timer/counter A1  
**DM7820\_ADVINT\_MASTER\_8254\_A\_2** 8254 timer/counter A2  
**DM7820\_ADVINT\_MASTER\_8254\_B\_0** 8254 timer/counter B0  
**DM7820\_ADVINT\_MASTER\_8254\_B\_1** 8254 timer/counter B1  
**DM7820\_ADVINT\_MASTER\_8254\_B\_2** 8254 timer/counter B2  
**DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_0** Programmable clock 0  
**DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_1** Programmable clock 1  
**DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_2** Programmable clock 2  
**DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_3** Programmable clock 3  
**DM7820\_ADVINT\_MASTER\_STROBE\_1** Strobe signal 1  
**DM7820\_ADVINT\_MASTER\_STROBE\_2** Strobe signal 2  
**DM7820\_ADVINT\_MASTER\_INV\_STROBE\_1** Inverted strobe signal 1  
**DM7820\_ADVINT\_MASTER\_INV\_STROBE\_2** Inverted strobe signal 2

Definition at line 2674 of file dm7820\_types.h.

#### 4.63.2.3 enum\_dm7820\_advint\_mode

Advanced interrupt modes.

Enumerator:

**DM7820\_ADVINT\_MODE\_DISABLED** Disabled  
**DM7820\_ADVINT\_MODE\_STROBE** Strobe mode  
**DM7820\_ADVINT\_MODE\_MATCH** Match mode  
**DM7820\_ADVINT\_MODE\_EVENT** Event mode

Definition at line 2635 of file dm7820\_types.h.



## 4.64 DM7820 type interrupt enumerations

### Data Structures

- struct `_dm7820_interrupt_info`  
*Interrupt source information.*

### Typedefs

- typedef enum  
`_dm7820_interrupt_source` `dm7820_interrupt_source`  
*Interrupt source type.*
- typedef enum  
`_dm7820_minor_interrupt_register` `dm7820_minor_interrupt_register`  
*Minor interrupt control/status register type.*
- typedef struct  
`_dm7820_interrupt_info` `dm7820_interrupt_info`  
*Interrupt source information type.*

### Enumerations

- enum `_dm7820_interrupt_source` {  
`DM7820_INTERRUPT_ADVINT_0 = 0, DM7820_INTERRUPT_ADVINT_1, DM7820_INTERRUPT_FIFO_0_EMPTY, DM7820_INTERRUPT_FIFO_0_FULL,`  
`DM7820_INTERRUPT_FIFO_0_OVERFLOW, DM7820_INTERRUPT_FIFO_0_READ_REQUEST, DM7820_INTERRUPT_FIFO_0_UNDERFLOW, DM7820_INTERRUPT_FIFO_0_WRITE_REQUEST,`  
`DM7820_INTERRUPT_FIFO_1_EMPTY, DM7820_INTERRUPT_FIFO_1_FULL, DM7820_INTERRUPT_FIFO_1_OVERFLOW, DM7820_INTERRUPT_FIFO_1_READ_REQUEST,`  
`DM7820_INTERRUPT_FIFO_1_UNDERFLOW, DM7820_INTERRUPT_FIFO_1_WRITE_REQUEST, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_A_POSITIVE_ROLLOVER,`  
`DM7820_INTERRUPT_INCENC_0_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_0_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_A_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_A_POSITIVE_ROLLOVER,`  
`DM7820_INTERRUPT_INCENC_1_CHANNEL_B_NEGATIVE_ROLLOVER, DM7820_INTERRUPT_INCENC_1_CHANNEL_B_POSITIVE_ROLLOVER, DM7820_INTERRUPT_PRGCLK_0, DM7820_INTERRUPT_PRGCLK_1,`  
`DM7820_INTERRUPT_PRGCLK_2, DM7820_INTERRUPT_PRGCLK_3, DM7820_INTERRUPT_PWM_0, DM7820_INTERRUPT_PWM_1,`  
`DM7820_INTERRUPT_TMRCTR_A_0, DM7820_INTERRUPT_TMRCTR_A_1, DM7820_INTERRUPT_TMRCTR_A_2, DM7820_INTERRUPT_TMRCTR_B_0,`  
`DM7820_INTERRUPT_TMRCTR_B_1, DM7820_INTERRUPT_TMRCTR_B_2, DM7820_INTERRUPT_FIFO_0_DMA_DONE, DM7820_INTERRUPT_FIFO_1_DMA_DONE,`  
`DM7820_INTERRUPT_NONE` }  
*Interrupt sources.*
- enum `_dm7820_minor_interrupt_register` {  
`DM7820_MINOR_INT_REG_FIFO_0_INT = 0, DM7820_MINOR_INT_REG_FIFO_1_INT, DM7820_MINOR_INT_REG_INCENC_0_INT, DM7820_MINOR_INT_REG_INCENC_1_INT,`  
`DM7820_MINOR_INT_REG_TMRCTR_INT, DM7820_MINOR_INT_REG_NONE` }  
*Minor interrupt control/status registers.*

#### 4.64.1 Detailed Description

DM7820\_Types\_AdvInt\_Enumerations

## 4.64.2 Enumeration Type Documentation

### 4.64.2.1 enum\_dm7820\_interrupt\_source

Interrupt sources.

Enumerator:

**DM7820\_INTERRUPT\_ADVINT\_0** Advanced interrupt block 0 interrupt

**DM7820\_INTERRUPT\_ADVINT\_1** Advanced interrupt block 1 interrupt (1)

**DM7820\_INTERRUPT\_FIFO\_0\_EMPTY** FIFO block FIFO 0 empty interrupt (2)

**DM7820\_INTERRUPT\_FIFO\_0\_FULL** FIFO block FIFO 0 full interrupt (3)

**DM7820\_INTERRUPT\_FIFO\_0\_OVERFLOW** FIFO block FIFO 0 overflow interrupt (4)

**DM7820\_INTERRUPT\_FIFO\_0\_READ\_REQUEST** FIFO block FIFO 0 read request interrupt (5)

**DM7820\_INTERRUPT\_FIFO\_0\_UNDERFLOW** FIFO block FIFO 0 underflow interrupt (6)

**DM7820\_INTERRUPT\_FIFO\_0\_WRITE\_REQUEST** FIFO block FIFO 0 write request interrupt (7)

**DM7820\_INTERRUPT\_FIFO\_1\_EMPTY** FIFO block FIFO 1 empty interrupt (8)

**DM7820\_INTERRUPT\_FIFO\_1\_FULL** FIFO block FIFO 1 full interrupt (9)

**DM7820\_INTERRUPT\_FIFO\_1\_OVERFLOW** FIFO block FIFO 1 overflow interrupt (10)

**DM7820\_INTERRUPT\_FIFO\_1\_READ\_REQUEST** FIFO block FIFO 1 read request interrupt (11)

**DM7820\_INTERRUPT\_FIFO\_1\_UNDERFLOW** FIFO block FIFO 1 underflow interrupt (12)

**DM7820\_INTERRUPT\_FIFO\_1\_WRITE\_REQUEST** FIFO block FIFO 1 write request interrupt (13)

**DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_A\_NEGATIVE\_ROLLOVER** Incremental encoder block 0 channel A negative rollover interrupt (14)

**DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_A\_POSITIVE\_ROLLOVER** Incremental encoder block 0 channel A positive rollover interrupt (15)

**DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_B\_NEGATIVE\_ROLLOVER** Incremental encoder block 0 channel B negative rollover interrupt (16)

**DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_B\_POSITIVE\_ROLLOVER** Incremental encoder block 0 channel B positive rollover interrupt (17)

**DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_A\_NEGATIVE\_ROLLOVER** Incremental encoder block 1 channel A negative rollover interrupt (18)

**DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_A\_POSITIVE\_ROLLOVER** Incremental encoder block 1 channel A positive rollover interrupt (19)

**DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_B\_NEGATIVE\_ROLLOVER** Incremental encoder block 1 channel B negative rollover interrupt (20)

**DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_B\_POSITIVE\_ROLLOVER** Incremental encoder block 1 channel B positive rollover interrupt (21)

**DM7820\_INTERRUPT\_PRGCLK\_0** Programmable clock block 0 interrupt (22)

**DM7820\_INTERRUPT\_PRGCLK\_1** Programmable clock block 1 interrupt (23)

**DM7820\_INTERRUPT\_PRGCLK\_2** Programmable clock block 2 interrupt (24)

**DM7820\_INTERRUPT\_PRGCLK\_3** Programmable clock block 3 interrupt (25)

**DM7820\_INTERRUPT\_PWM\_0** Pulse width modulator block 0 interrupt (26)

**DM7820\_INTERRUPT\_PWM\_1** Pulse width modulator block 1 interrupt (27)

**DM7820\_INTERRUPT\_TMRCTR\_A\_0** 8254 timer/counter A0 interrupt (28)

**DM7820\_INTERRUPT\_TMRCTR\_A\_1** 8254 timer/counter A1 interrupt (29)

**DM7820\_INTERRUPT\_TMRCTR\_A\_2** 8254 timer/counter A2 interrupt (30)

**DM7820\_INTERRUPT\_TMRCTR\_B\_0** 8254 timer/counter B0 interrupt (31)

**DM7820\_INTERRUPT\_TMRCTR\_B\_1** 8254 timer/counter B1 interrupt (32)

**DM7820\_INTERRUPT\_TMRCTR\_B\_2** 8254 timer/counter B2 interrupt (33)

**DM7820\_INTERRUPT\_FIFO\_0\_DMA\_DONE** FIFO block FIFO 0 DMA done interrupt. Applications cannot control this interrupt but they can get its status. (34)

**DM7820\_INTERRUPT\_FIFO\_1\_DMA\_DONE** FIFO block FIFO 1 DMA done interrupt. Applications cannot control this interrupt but they can get its status. (35)

**DM7820\_INTERRUPT\_NONE** Value which indicates no interrupt source. User level ignores this. The kernel uses this in the interrupt handler. This must be the last entry. (36)

Definition at line 2798 of file dm7820\_types.h.

#### 4.64.2.2 enum\_dm7820\_minor\_interrupt\_register

Minor interrupt control/status registers.

Enumerator:

**DM7820\_MINOR\_INT\_REG\_FIFO\_0\_INT** FIFO 0 Interrupt Register

**DM7820\_MINOR\_INT\_REG\_FIFO\_1\_INT** FIFO 1 Interrupt Register

**DM7820\_MINOR\_INT\_REG\_INCENC\_0\_INT** Incremental Encoder 0 Interrupt Register

**DM7820\_MINOR\_INT\_REG\_INCENC\_1\_INT** Incremental Encoder 1 Interrupt Register

**DM7820\_MINOR\_INT\_REG\_TMRCTR\_INT** 8254 Timer/Counter Interrupt Register

**DM7820\_MINOR\_INT\_REG\_NONE** Value which indicates no minor interrupt register. This must be the last entry.

Definition at line 3038 of file dm7820\_types.h.

## 4.65 DM7820 type definition structures

### Data Structures

- struct [dm7820\\_pci\\_access\\_request](#)  
*PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.*
- struct [dm7820\\_interrupt\\_control](#)  
*Structure containing information needed to acknowledge, disable, and enable a particular interrupt source.*
- struct [dm7820\\_minor\\_int\\_reg\\_layout](#)  
*Minor interrupt register bit layout.*

### Typedefs

- typedef struct  
[dm7820\\_pci\\_access\\_request dm7820\\_pci\\_access\\_request\\_t](#)
- typedef struct  
[dm7820\\_interrupt\\_control dm7820\\_interrupt\\_control\\_t](#)  
*Interrupt control information type.*
- typedef struct  
[dm7820\\_minor\\_int\\_reg\\_layout dm7820\\_minor\\_int\\_reg\\_layout\\_t](#)  
*Minor interrupt register bit layout type.*

#### 4.65.1 Detailed Description

DM7820\_Types\_Enumerations

#### 4.65.2 Typedef Documentation

##### 4.65.2.1 typedef struct dm7820\_pci\_access\_request dm7820\_pci\_access\_request\_t

PCI region access request descriptor type

Definition at line 3191 of file dm7820\_types.h.

## 4.66 DM7820 type definition typedefs

### Typedefs

- typedef uint64\_t [dm7820\\_int\\_source\\_status\\_t](#)  
*Interrupt source status type.*

### 4.66.1 Detailed Description

DM7820\_Types\_Structures



## Chapter 5

# Data Structure Documentation

### 5.1 `_dm7820_interrupt_info` Struct Reference

Interrupt source information.

```
#include <dm7820_types.h>
```

#### Data Fields

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

#### 5.1.1 Detailed Description

Interrupt source information.

Definition at line 3091 of file `dm7820_types.h`.

#### 5.1.2 Field Documentation

##### 5.1.2.1 `int error`

Error Code: 0 = success, -1 = failure

Definition at line 3107 of file `dm7820_types.h`.

Referenced by `ISR()`.

##### 5.1.2.2 `int int_missed`

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

Definition at line 3103 of file `dm7820_types.h`.

##### 5.1.2.3 `int int_remaining`

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

Definition at line 3099 of file `dm7820_types.h`.

#### 5.1.2.4 dm7820\_interrupt\_source source

The interrupt sources for the last acknowledged interrupt

Definition at line 3095 of file dm7820\_types.h.

Referenced by ISR(), and main().

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

- include/dm7820\_types.h

## 5.2 DM7820\_Board\_Descriptor Struct Reference

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

```
#include <dm7820_library.h>
```

### Data Fields

- int [file\\_descriptor](#)
- void(\* [isr](#))(dm7820\_interrupt\_info status)
- pthread\_t [pid](#)

#### 5.2.1 Detailed Description

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

Definition at line 214 of file dm7820\_library.h.

#### 5.2.2 Field Documentation

##### 5.2.2.1 int file\_descriptor

File descriptor for device returned from open()

Definition at line 220 of file dm7820\_library.h.

Referenced by main().

##### 5.2.2.2 void(\* isr)(dm7820\_interrupt\_info status)

Function pointer to the user ISR callback function.

Definition at line 226 of file dm7820\_library.h.

##### 5.2.2.3 pthread\_t pid

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

Definition at line 232 of file dm7820\_library.h.

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

- include/dm7820\_library.h



## 5.3 dm7820\_device\_descriptor Struct Reference

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

```
#include <dm7820_driver.h>
```

### Data Fields

- char [device\\_name](#) [DM7820\_DEVICE\_NAME\_LENGTH]
- [dm7820\\_pci\\_region\\_t](#) [pci](#) [PCI\_ROM\_RESOURCE]
- [spinlock\\_t](#) [device\\_lock](#)
- [uint8\\_t](#) [reference\\_count](#)
- unsigned int [irq\\_number](#)
- [uint8\\_t](#) [interrupt\\_occurred](#)
- [uint8\\_t](#) [remove\\_isr\\_flag](#)
- [wait\\_queue\\_head\\_t](#) [int\\_wait\\_queue](#)
- [wait\\_queue\\_head\\_t](#) [dma\\_wait\\_queue](#)
- [dm7820\\_int\\_source\\_status\\_t](#) [int\\_source\\_status](#)
- [uint8\\_t](#) [dma\\_initialized](#) [DM7820\_FIFO\_CHANNELS]
- [uint32\\_t](#) [dma\\_size](#) [DM7820\_FIFO\_CHANNELS]
- [struct list\\_head](#) [dma\\_buffers\\_pre\\_transfer](#) [DM7820\_FIFO\_CHANNELS]
- [struct list\\_head](#) [dma\\_buffers\\_post\\_transfer](#) [DM7820\_FIFO\_CHANNELS]
- [uint8\\_t](#) [dma\\_in\\_read\\_direction](#) [DM7820\_FIFO\_CHANNELS]
- [dm7820\\_interrupt\\_source](#) [int\\_status](#) [DM7820\_INT\_QUEUE\_SIZE]
- unsigned int [int\\_queue\\_in](#)
- unsigned int [int\\_queue\\_out](#)
- unsigned int [int\\_queue\\_missed](#)
- unsigned int [int\\_queue\\_count](#)

### 5.3.1 Detailed Description

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

Definition at line 305 of file `dm7820_driver.h`.

### 5.3.2 Field Documentation

#### 5.3.2.1 [spinlock\\_t](#) [device\\_lock](#)

Concurrency control

Definition at line 324 of file `dm7820_driver.h`.

#### 5.3.2.2 [char](#) [device\\_name](#)[DM7820\_DEVICE\_NAME\_LENGTH]

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

Definition at line 312 of file `dm7820_driver.h`.

#### 5.3.2.3 [struct list\\_head](#) [dma\\_buffers\\_post\\_transfer](#)[DM7820\_FIFO\_CHANNELS]

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

Definition at line 397 of file `dm7820_driver.h`.

#### 5.3.2.4 struct list\_head dma\_buffers\_pre\_transfer[DM7820\_FIFO\_CHANNELS]

Per-FIFO channel linked list of DMA buffers

Definition at line 390 of file dm7820\_driver.h.

#### 5.3.2.5 uint8\_t dma\_in\_read\_direction[DM7820\_FIFO\_CHANNELS]

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

Definition at line 404 of file dm7820\_driver.h.

#### 5.3.2.6 uint8\_t dma\_initialized[DM7820\_FIFO\_CHANNELS]

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

Definition at line 378 of file dm7820\_driver.h.

#### 5.3.2.7 uint32\_t dma\_size[DM7820\_FIFO\_CHANNELS]

Per-FIFO channel DMA transfer size

Definition at line 384 of file dm7820\_driver.h.

#### 5.3.2.8 wait\_queue\_head\_t dma\_wait\_queue

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

Definition at line 362 of file dm7820\_driver.h.

#### 5.3.2.9 unsigned int int\_queue\_count

Number of interrupts currently in the queue

Definition at line 434 of file dm7820\_driver.h.

#### 5.3.2.10 unsigned int int\_queue\_in

Number of entries in the interrupt status queue

Definition at line 416 of file dm7820\_driver.h.

#### 5.3.2.11 unsigned int int\_queue\_missed

Number of interrupts missed because of a full queue

Definition at line 428 of file dm7820\_driver.h.

#### 5.3.2.12 unsigned int int\_queue\_out

Number of entries read from the interrupt status queue

Definition at line 422 of file dm7820\_driver.h.

#### 5.3.2.13 dm7820\_int\_source\_status\_t int\_source\_status

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

Definition at line 370 of file dm7820\_driver.h.

#### 5.3.2.14 dm7820\_interrupt\_source int.status[DM7820\_INT\_QUEUE\_SIZE]

Interrupt status queue

Definition at line 410 of file dm7820\_driver.h.

#### 5.3.2.15 wait\_queue\_head\_t int.wait\_queue

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

Definition at line 356 of file dm7820\_driver.h.

#### 5.3.2.16 uint8\_t interrupt\_occurred

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

Definition at line 344 of file dm7820\_driver.h.

#### 5.3.2.17 unsigned int irq\_number

IRQ line number

Definition at line 337 of file dm7820\_driver.h.

#### 5.3.2.18 dm7820\_pci\_region\_t pci[PCI\_ROM\_RESOURCE]

Information about each of the standard PCI regions

Definition at line 318 of file dm7820\_driver.h.

#### 5.3.2.19 uint8\_t reference\_count

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

Definition at line 331 of file dm7820\_driver.h.

#### 5.3.2.20 uint8\_t remove\_isr\_flag

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

Definition at line 350 of file dm7820\_driver.h.

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

- [include/dm7820\\_driver.h](#)

## 5.4 dm7820\_dma\_descriptor\_t Struct Reference

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

```
#include <dm7820_driver.h>
```

## Data Fields

- `dma_addr_t` [bus\\_address](#)
- `void *` [virtual\\_address](#)

### 5.4.1 Detailed Description

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

Definition at line 264 of file `dm7820_driver.h`.

### 5.4.2 Field Documentation

#### 5.4.2.1 `dma_addr_t bus_address`

Bus/physical address

Definition at line 270 of file `dm7820_driver.h`.

#### 5.4.2.2 `void* virtual_address`

Virtual address

Definition at line 276 of file `dm7820_driver.h`.

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

- `include/dm7820_driver.h`

## 5.5 dm7820\_dma\_function\_arguments Union Reference

Structure encapsulating arguments to all possible DMA functions.

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

## Data Fields

- [dm7820\\_dma\\_initialize\\_arguments\\_t dma\\_init](#)

### 5.5.1 Detailed Description

Structure encapsulating arguments to all possible DMA functions.

Definition at line 236 of file `dm7820_ioctl.h`.

### 5.5.2 Field Documentation

#### 5.5.2.1 `dm7820_dma_initialize_arguments_t dma_init`

DMA initialization

Definition at line 242 of file `dm7820_ioctl.h`.

Referenced by `main()`.

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

- `include/dm7820_ioctl.h`

## 5.6 dm7820\_dma\_initialize\_arguments Struct Reference

Arguments for DMA initialization function.

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

### Data Fields

- `uint32_t buffer_count`
- `uint32_t buffer_size`

### 5.6.1 Detailed Description

Arguments for DMA initialization function.

Definition at line 216 of file `dm7820_ioctl.h`.

### 5.6.2 Field Documentation

#### 5.6.2.1 `uint32_t buffer_count`

Number of DMA buffers to allocate

Definition at line 222 of file `dm7820_ioctl.h`.

Referenced by `main()`.

#### 5.6.2.2 `uint32_t buffer_size`

DMA buffer size in bytes

Definition at line 228 of file `dm7820_ioctl.h`.

Referenced by `main()`.

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

- `include/dm7820_ioctl.h`

## 5.7 dm7820\_dma\_list\_item\_t Struct Reference

DM7820 DMA buffer list item.

```
#include <dm7820_driver.h>
```

### Data Fields

- `struct list_head list`
- `dm7820_dma_descriptor_t * dma_buffer`

### 5.7.1 Detailed Description

DM7820 DMA buffer list item.

Definition at line 284 of file dm7820\_driver.h.

### 5.7.2 Field Documentation

#### 5.7.2.1 dm7820\_dma\_descriptor\_t\* dma\_buffer

DMA buffer allocation information

Definition at line 296 of file dm7820\_driver.h.

#### 5.7.2.2 struct list\_head list

Linked list management

Definition at line 290 of file dm7820\_driver.h.

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

- [include/dm7820\\_driver.h](#)

## 5.8 dm7820\_interrupt\_control Struct Reference

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

```
#include <dm7820_types.h>
```

### Data Fields

- [uint16\\_t int\\_enable\\_bit](#)
- [uint16\\_t int\\_status\\_bit](#)
- [dm7820\\_minor\\_interrupt\\_register minor\\_reg](#)
- [uint16\\_t minor\\_enable](#)
- [uint16\\_t minor\\_status](#)

### 5.8.1 Detailed Description

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

Definition at line 3199 of file dm7820\_types.h.

### 5.8.2 Field Documentation

#### 5.8.2.1 uint16\_t int\_enable\_bit

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

Definition at line 3206 of file dm7820\_types.h.

### 5.8.2.2 uint16\_t int\_status\_bit

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

Definition at line 3213 of file dm7820\_types.h.

### 5.8.2.3 uint16\_t minor\_enable

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

Definition at line 3226 of file dm7820\_types.h.

### 5.8.2.4 dm7820\_minor\_interrupt\_register minor\_reg

Minor interrupt register

Definition at line 3219 of file dm7820\_types.h.

### 5.8.2.5 uint16\_t minor\_status

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

Definition at line 3233 of file dm7820\_types.h.

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

- [include/dm7820\\_types.h](#)

## 5.9 dm7820\_interrupt\_status\_source Struct Reference

Interrupt source information for a single Interrupt Status Register bit.

```
#include <dm7820_driver.h>
```

### Data Fields

- [dm7820\\_minor\\_interrupt\\_register minor\\_reg](#)
- [dm7820\\_interrupt\\_source source](#)
- [dm7820\\_interrupt\\_source \\* source\\_table](#)

### 5.9.1 Detailed Description

Interrupt source information for a single Interrupt Status Register bit.

Definition at line 449 of file dm7820\_driver.h.

### 5.9.2 Field Documentation

#### 5.9.2.1 dm7820\_minor\_interrupt\_register minor\_reg

Minor interrupt register. If there is no minor interrupt register, this will be DM7820\_MINOR\_INT\_REG\_NONE.

Definition at line 456 of file dm7820\_driver.h.

### 5.9.2.2 dm7820\_interrupt\_source source

Interrupt source for register bit. If there is a minor interrupt register, this will be DM7820\_INTERRUPT\_NONE.

Definition at line 463 of file dm7820\_driver.h.

### 5.9.2.3 dm7820\_interrupt\_source\* source\_table

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

Definition at line 470 of file dm7820\_driver.h.

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

- [include/dm7820\\_driver.h](#)

## 5.10 dm7820\_ioctl\_argument Union Reference

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

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

### Data Fields

- [dm7820\\_ioctl\\_region\\_readwrite\\_t](#) readwrite
- [dm7820\\_ioctl\\_region\\_modify\\_t](#) modify
- [dm7820\\_ioctl\\_interrupt\\_status\\_t](#) int\_status
- [dm7820\\_ioctl\\_dma\\_function\\_t](#) dma\_function

### 5.10.1 Detailed Description

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

Definition at line 307 of file dm7820\_ioctl.h.

### 5.10.2 Field Documentation

#### 5.10.2.1 dm7820\_ioctl\_dma\_function\_t dma\_function

DMA management function

Definition at line 331 of file dm7820\_ioctl.h.

Referenced by main().

#### 5.10.2.2 dm7820\_ioctl\_interrupt\_status\_t int\_status

Get interrupt status

Definition at line 325 of file dm7820\_ioctl.h.

Referenced by main().



#### 5.10.2.3 dm7820\_ioctl\_region\_modify\_t modify

PCI region read/modify/write

Definition at line 319 of file dm7820\_ioctl.h.

Referenced by main().

#### 5.10.2.4 dm7820\_ioctl\_region\_readwrite\_t readwrite

PCI region read and write

Definition at line 313 of file dm7820\_ioctl.h.

Referenced by main().

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

- [include/dm7820\\_ioctl.h](#)

## 5.11 dm7820\_ioctl\_dma\_function Struct Reference

ioctl() request structure for performing a DMA function

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

### Data Fields

- void \* [user\\_buffer](#)
- uint8\_t [direction](#)
- uint32\_t [DMA\\_Transfer\\_Size](#)
- uint32\_t [buffer\\_address](#)
- uint32\_t [transfer\\_size](#)
- [dm7820\\_fifo\\_queue](#) fifo
- [dm7820\\_dma\\_manage\\_function\\_t](#) function
- [dm7820\\_dma\\_function\\_arguments\\_t](#) arguments

### 5.11.1 Detailed Description

ioctl() request structure for performing a DMA function

Definition at line 250 of file dm7820\_ioctl.h.

### 5.11.2 Field Documentation

#### 5.11.2.1 dm7820\_dma\_function\_arguments\_t arguments

Arguments required by function

Definition at line 298 of file dm7820\_ioctl.h.

Referenced by main().

#### 5.11.2.2 uint32\_t buffer\_address

The address of a DMA buffer

Definition at line 274 of file dm7820\_ioctl.h.

#### 5.11.2.3 `uint8_t direction`

enumeration of the direction mode for the DMA channel

Definition at line 262 of file `dm7820_ioctl.h`.

#### 5.11.2.4 `uint32_t DMA_Transfer_Size`

contains the transfer size for the DMA channel

Definition at line 268 of file `dm7820_ioctl.h`.

#### 5.11.2.5 `dm7820_fifo_queue fifo`

DMA/FIFO channel to operate upon

Definition at line 286 of file `dm7820_ioctl.h`.

Referenced by `main()`.

#### 5.11.2.6 `dm7820_dma_manage_function_t function`

DMA function to perform

Definition at line 292 of file `dm7820_ioctl.h`.

Referenced by `main()`.

#### 5.11.2.7 `uint32_t transfer_size`

Size of the DMA transfer

Definition at line 280 of file `dm7820_ioctl.h`.

Referenced by `main()`.

#### 5.11.2.8 `void* user_buffer`

Buffer for data coming from user.

Definition at line 256 of file `dm7820_ioctl.h`.

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

- [include/dm7820\\_ioctl.h](#)

## 5.12 `dm7820_ioctl_interrupt_status` Struct Reference

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

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

### Data Fields

- `uint8_t wait_for_interrupt`
- `dm7820_interrupt_info int_source_info`

### 5.12.1 Detailed Description

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

Definition at line 183 of file dm7820\_ioctl.h.

### 5.12.2 Field Documentation

#### 5.12.2.1 dm7820\_interrupt\_info int\_source\_info

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

Definition at line 200 of file dm7820\_ioctl.h.

#### 5.12.2.2 uint8\_t wait\_for\_interrupt

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

Definition at line 192 of file dm7820\_ioctl.h.

Referenced by main().

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

- [include/dm7820\\_ioctl.h](#)

## 5.13 dm7820\_ioctl\_region\_modify Struct Reference

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

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

### Data Fields

- [dm7820\\_pci\\_access\\_request\\_t access](#)
- union {
  - [uint8\\_t mask8](#)
  - [uint16\\_t mask16](#)
  - [uint32\\_t mask32](#)
- [mask](#)

### 5.13.1 Detailed Description

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

Definition at line 128 of file dm7820\_ioctl.h.

### 5.13.2 Field Documentation

#### 5.13.2.1 dm7820\_pci\_access\_request\_t access

PCI region access request

Definition at line 134 of file dm7820\_ioctl.h.

Referenced by main().

#### 5.13.2.2 union { ... } mask

Bit mask that controls which bits can be modified. A zero in a bit position means that the corresponding register bit should not be modified. A one in a bit position means that the corresponding register bit should be modified.

Note that it's possible to set bits outside of the mask depending upon the register value before modification. When processing the associated request code, the driver will silently prevent this from happening but will not return an indication that the mask or new value was incorrect.

Referenced by main().

#### 5.13.2.3 uint16\_t mask16

Mask for 16-bit operations

Definition at line 160 of file dm7820\_ioctl.h.

Referenced by main().

#### 5.13.2.4 uint32\_t mask32

Mask for 32-bit operations

Definition at line 166 of file dm7820\_ioctl.h.

Referenced by main().

#### 5.13.2.5 uint8\_t mask8

Mask for 8-bit operations

Definition at line 154 of file dm7820\_ioctl.h.

Referenced by main().

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

- [include/dm7820\\_ioctl.h](#)

## 5.14 dm7820\_ioctl\_region\_readwrite Struct Reference

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

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

### Data Fields

- [dm7820\\_pci\\_access\\_request\\_t access](#)

#### 5.14.1 Detailed Description

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

Definition at line 108 of file dm7820\_ioctl.h.

### 5.14.2 Field Documentation

#### 5.14.2.1 dm7820\_pci\_access\_request\_t access

PCI region access request

Definition at line 114 of file dm7820\_ioctl.h.

Referenced by main().

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

- include/dm7820\_ioctl.h

## 5.15 dm7820\_minor\_int\_reg\_layout Struct Reference

Minor interrupt register bit layout.

```
#include <dm7820_types.h>
```

### Data Fields

- uint16\_t [offset](#)
- uint16\_t [enable\\_mask](#)
- uint16\_t [reserved\\_mask](#)
- uint16\_t [status\\_mask](#)

### 5.15.1 Detailed Description

Minor interrupt register bit layout.

Definition at line 3248 of file dm7820\_types.h.

### 5.15.2 Field Documentation

#### 5.15.2.1 uint16\_t enable\_mask

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

Definition at line 3263 of file dm7820\_types.h.

#### 5.15.2.2 uint16\_t offset

BAR2 register offset

Definition at line 3254 of file dm7820\_types.h.

#### 5.15.2.3 uint16\_t reserved\_mask

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

Definition at line 3271 of file dm7820\_types.h.

#### 5.15.2.4 uint16\_t status\_mask

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

Definition at line 3280 of file dm7820\_types.h.

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

- [include/dm7820\\_types.h](#)

## 5.16 dm7820\_pci\_access\_request Struct Reference

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

```
#include <dm7820_types.h>
```

### Data Fields

- [dm7820\\_pci\\_region\\_access\\_size\\_t](#) size
- [dm7820\\_pci\\_region\\_num\\_t](#) region
- [uint16\\_t](#) offset
- union {
  - [uint8\\_t](#) data8
  - [uint16\\_t](#) data16
  - [uint32\\_t](#) data32
- } data

### 5.16.1 Detailed Description

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

Definition at line 3141 of file dm7820\_types.h.

### 5.16.2 Field Documentation

#### 5.16.2.1 union { ... } data

Data to write or the data read

Referenced by `main()`.

#### 5.16.2.2 uint16\_t data16

16-bit value

Definition at line 3177 of file dm7820\_types.h.

Referenced by `main()`.

#### 5.16.2.3 uint32\_t data32

32-bit value

Definition at line 3183 of file dm7820\_types.h.

Referenced by main().

#### 5.16.2.4 uint8\_t data8

8-bit value

Definition at line 3171 of file dm7820\_types.h.

Referenced by main().

#### 5.16.2.5 uint16\_t offset

Offset within region to access

Definition at line 3159 of file dm7820\_types.h.

Referenced by main().

#### 5.16.2.6 dm7820\_pci\_region\_num\_t region

The PCI region to access

Definition at line 3153 of file dm7820\_types.h.

Referenced by main().

#### 5.16.2.7 dm7820\_pci\_region\_access\_size\_t size

Size of access in bits

Definition at line 3147 of file dm7820\_types.h.

Referenced by main().

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

- include/[dm7820\\_types.h](#)

## 5.17 dm7820\_pci\_region Struct Reference

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

```
#include <dm7820_driver.h>
```

### Data Fields

- unsigned long [io\\_addr](#)
- unsigned long [length](#)
- unsigned long [phys\\_addr](#)
- void \* [virt\\_addr](#)
- uint8\_t [allocated](#)

### 5.17.1 Detailed Description

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

Definition at line 214 of file `dm7820_driver.h`.

### 5.17.2 Field Documentation

#### 5.17.2.1 `uint8_t allocated`

Flag indicating whether or not the I/O-mapped memory ranged was allocated. A value of zero means the memory range was not allocated. Any other value means the memory range was allocated.

Definition at line 248 of file `dm7820_driver.h`.

#### 5.17.2.2 `unsigned long io_addr`

I/O port number if I/O mapped

Definition at line 220 of file `dm7820_driver.h`.

#### 5.17.2.3 `unsigned long length`

Length of region in bytes

Definition at line 226 of file `dm7820_driver.h`.

#### 5.17.2.4 `unsigned long phys_addr`

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

Definition at line 233 of file `dm7820_driver.h`.

#### 5.17.2.5 `void* virt_addr`

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

Definition at line 240 of file `dm7820_driver.h`.

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

- [include/dm7820\\_driver.h](#)

## 5.18 `register_read` Struct Reference

### Data Fields

- [dm7820\\_pci\\_region\\_access\\_size\\_t](#) `size`
- [dm7820\\_pci\\_region\\_num\\_t](#) `region`
- [uint16\\_t](#) `offset`
- [uint32\\_t](#) `expected`



### 5.18.1 Detailed Description

Register read test information

Definition at line 118 of file `basic_test.c`.

### 5.18.2 Field Documentation

#### 5.18.2.1 `uint32_t` expected

Expected value of read; recast based upon size of read

Definition at line 142 of file `basic_test.c`.

Referenced by `main()`.

#### 5.18.2.2 `uint16_t` offset

Offset within region to read

Definition at line 136 of file `basic_test.c`.

Referenced by `main()`.

#### 5.18.2.3 `dm7820_pci_region_num_t` region

The PCI region to read

Definition at line 130 of file `basic_test.c`.

Referenced by `main()`.

#### 5.18.2.4 `dm7820_pci_region_access_size_t` size

Size of read in bits

Definition at line 124 of file `basic_test.c`.

Referenced by `main()`.

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

- [examples/basic\\_test.c](#)



## Chapter 6

# File Documentation

### 6.1 examples/advint\_event\_interrupt.c File Reference

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

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

#### Functions

- void [ISR](#) ([dm7820\\_interrupt\\_info](#) interrupt\_info)  
*Userspace ISR.*
- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

#### Variables

- static char \* [program\\_name](#)
- volatile unsigned int [advint1\\_interrupts](#)

#### 6.1.1 Detailed Description

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

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

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

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

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

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

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

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

#### Note

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

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[advint\\_event\\_interrupt.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [advint\\_event\\_interrupt.c](#).

## 6.1.2 Function Documentation

### 6.1.2.1 void ISR ( dm7820\_interrupt\_info *interrupt\_info* )

Userspace ISR.

#### Parameters

<i>interrupt_info</i>	Information about the interrupt, returned by the kernel driver
-----------------------	--

Definition at line 98 of file [advint\\_event\\_interrupt.c](#).

References [advint1\\_interrupts](#), [DM7820\\_INTERRUPT\\_ADVINT\\_1](#), [DM7820\\_Return\\_Status](#), [\\_dm7820\\_interrupt\\_info::error](#), and [\\_dm7820\\_interrupt\\_info::source](#).

Referenced by [main\(\)](#).

### 6.1.2.2 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

**Parameters**

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

**Return values**

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 168 of file `advint_event_interrupt.c`.

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

**6.1.3 Variable Documentation****6.1.3.1 volatile unsigned int `advint1_interrupts`**

Counter for `advint` interrupts received

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

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

**6.1.3.2 char\* `program_name` [static]**

Name of the program as invoked on the command line

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

Referenced by `main()`, and `usage()`.

**6.2 examples/advint\_match\_interrupt.c File Reference**

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

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

**Macros**

- `#define COMPARE_REGISTER_VALUE 0x8000`

## Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)

### 6.2.1 Detailed Description

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

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

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

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

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

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[advint\\_match\\_interrupt.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [advint\\_match\\_interrupt.c](#).

### 6.2.2 Macro Definition Documentation

#### 6.2.2.1 #define COMPARE\_REGISTER\_VALUE 0x8000

Value to load into compare register

Definition at line 61 of file [advint\\_match\\_interrupt.c](#).

Referenced by [main\(\)](#).

### 6.2.3 Function Documentation

#### 6.2.3.1 int main ( int argument\_count, char \*\* arguments )

Main program code.

## Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

## Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 127 of file `advint_match_interrupt.c`.

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

## 6.2.4 Variable Documentation

### 6.2.4.1 `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 71 of file `advint_match_interrupt.c`.

Referenced by `main()`, and `usage()`.

## 6.3 examples/advint\_status.c File Reference

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

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

### Macros

- `#define COMPARE_REGISTER_VALUE 0x8000`

### Functions

- static void `usage` (void)

*Print information on stderr about how the program is to be used. After doing so, the program is exited.*

- `int main (int argument_count, char **arguments)`  
*Main program code.*

## Variables

- `static char * program_name`

### 6.3.1 Detailed Description

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

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

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

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

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[advint\\_status.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [advint\\_status.c](#).

### 6.3.2 Macro Definition Documentation

#### 6.3.2.1 `#define COMPARE_REGISTER_VALUE 0x8000`

Value to load into compare register

Definition at line 58 of file [advint\\_status.c](#).

Referenced by [main\(\)](#).

### 6.3.3 Function Documentation

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

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.



## Return values

<code>EXIT_SUCCESS</code>	Success.
<code>EXIT_FAILURE</code>	Failure.

Definition at line 124 of file `advint_status.c`.

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

### 6.3.4 Variable Documentation

#### 6.3.4.1 `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 68 of file `advint_status.c`.

Referenced by `main()`, and `usage()`.

## 6.4 examples/advint\_strobe\_interrupt.c File Reference

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

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

### Functions

- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* `program_name`

#### 6.4.1 Detailed Description

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

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

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

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

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

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

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

#### Note

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

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[advint\\_strobe\\_interrupt.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [advint\\_strobe\\_interrupt.c](#).

## 6.4.2 Function Documentation

### 6.4.2.1 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 125 of file [advint\\_strobe\\_interrupt.c](#).

References [board](#), [DM7820\\_ADVINT\\_INTERRUPT\\_0](#), [DM7820\\_ADVINT\\_INTERRUPT\\_1](#), [DM7820\\_ADVINT\\_MASTER\\_STROBE\\_1](#), [DM7820\\_ADVINT\\_MODE\\_DISABLED](#), [DM7820\\_ADVINT\\_MODE\\_STROBE](#), [DM7820\\_AdvInt\\_Read\\_Capture\(\)](#), [DM7820\\_AdvInt\\_Set\\_Master\(\)](#), [DM7820\\_AdvInt\\_Set\\_Mode\(\)](#), [DM7820\\_General\\_Close\\_Board\(\)](#), [DM7820\\_General\\_Enable\\_Interrupt\(\)](#), [DM7820\\_General\\_Get\\_Interrupt\\_Status\(\)](#), [DM7820\\_General\\_Open\\_Board\(\)](#), [DM7820\\_General\\_Reset\(\)](#), [DM7820\\_INTERRUPT\\_ADVINT\\_0](#), [DM7820\\_Return\\_Status](#), [D-](#)

M7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_MODE\_OUTPUT, DM7820\_STDIO\_PORT\_0, DM7820\_STDIO\_PORT\_1, DM7820\_StdIO\_Set\_IO\_Mode(), DM7820\_StdIO\_Set\_Output(), DM7820\_STDIO\_STROBE\_1, DM7820\_STDIO\_STROBE\_2, DM7820\_StdIO\_Strobe\_Mode(), DM7820\_StdIO\_Strobe\_Output(), program\_name, \_dm7820\_interrupt\_info::source, and usage().

### 6.4.3 Variable Documentation

#### 6.4.3.1 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 69 of file advint\_strobe\_interrupt.c.

Referenced by main(), and usage().

## 6.5 examples/basic\_test.c File Reference

Program which tests the basic functionality of the driver.

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

### Data Structures

- struct [register\\_read](#)

### Macros

- #define [DM7820\\_MAX\\_DMA\\_BUFFER\\_COUNT](#) 32
- #define [DM7820\\_MAX\\_DMA\\_BUFFER\\_SIZE](#) 0x40000

### Typedefs

- typedef struct [register\\_read](#) [register\\_read\\_t](#)
- typedef enum [initialization\\_state](#) [initialization\\_state\\_t](#)

## Enumerations

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

## Functions

- static void `cleanup` (void)  
*Perform actions necessary to clean up after the program encounters a fatal error.*
- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- static void `expect_success` (int status)  
*Verify that a function being tested succeeded.*
- static void `expect_failure_and_check` (int status, int expected\_errno)  
*Verify that a function being tested failed with the given errno.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* `program_name`
- static uint8\_t `access_sizes` []
- static uint16\_t `region_sizes` []
- static char \* `region_names` []
- `register_read_t` `register_reads` []
- static volatile `initialization_state_t` `init_state` = `INIT_NO_INITIALIZATION`
- static volatile int \* `descriptors`
- static volatile char `bad_device_name` [30]
- static volatile unsigned long `device_count` = 1

### 6.5.1 Detailed Description

Program which tests the basic functionality of the driver.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

## Warning

This program ABSOLUTELY IS NOT INTENDED to be an example of how to program a board. Some of the techniques appearing herein can lead to erratic program or system behavior and are used only to cause specific error conditions.

This program uses `sbrk()` to determine what should be an invalid address for causing certain errors in the driver. The address returned by `sbrk()` may not cause failure in some circumstances but there seems to be no reliable and easy way to determine whether an arbitrary user address is actually mapped into a process' address space.

Id:

[basic\\_test.c](#) 89872 2015-07-08 16:05:17Z rgroner

Definition in file [basic\\_test.c](#).

## 6.5.2 Macro Definition Documentation

### 6.5.2.1 `#define DM7820_MAX_DMA_BUFFER_COUNT 32`

Default driver limit on number of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 64 of file [basic\\_test.c](#).

Referenced by [main\(\)](#).

### 6.5.2.2 `#define DM7820_MAX_DMA_BUFFER_SIZE 0x40000`

Default driver limit in bytes on size of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 71 of file [basic\\_test.c](#).

Referenced by [main\(\)](#).

## 6.5.3 Typedef Documentation

### 6.5.3.1 `typedef enum initialization_state initialization_state_t`

Program initialization status type

Definition at line 159 of file [basic\\_test.c](#).

### 6.5.3.2 `typedef struct register_read register_read_t`

Register read test case type

Definition at line 153 of file [basic\\_test.c](#).

## 6.5.4 Enumeration Type Documentation

### 6.5.4.1 `enum initialization_state`

Status of program initialization. Used to indication which operations need to be undone when cleaning up after an error. This is a bit mask, so only one bit should be set for each enumeration value.

Enumerator:

**INIT\_NO\_INITIALIZATION** No initialization performed  
**INIT\_BOARD\_ARRAY\_ALLOCATED** File descriptor array allocated  
**INIT\_BAD\_DEVICE\_FILE\_CREATED** /dev entry with invalid minor number created  
**INIT\_BOARD\_ARRAY\_OPENED** File descriptor array contains at least one opened file  
**INIT\_NO\_INITIALIZATION** No initialization performed  
**INIT\_BOARD\_ARRAY\_ALLOCATED** Board descriptor array allocated  
**INIT\_BAD\_DEVICE\_FILE\_CREATED** /dev entry with invalid minor number created  
**INIT\_BOARD\_ARRAY\_OPENED** Board descriptor array contains at least one opened file

Definition at line 83 of file [basic\\_test.c](#).

### 6.5.5 Function Documentation

#### 6.5.5.1 static void expect\_failure\_and\_check ( int *status*, int *expected\_errno* ) [static]

Verify that a function being tested failed with the given errno.

##### Parameters

<i>status</i>	Return code from function being tested.
<i>expected_errno</i>	Expected errno that function should have set.

Definition at line 651 of file basic\_test.c.

Referenced by main().

#### 6.5.5.2 static void expect\_success ( int *status* ) [static]

Verify that a function being tested succeeded.

##### Parameters

<i>status</i>	Return code from function being tested.
---------------	---

Definition at line 624 of file basic\_test.c.

Referenced by main().

#### 6.5.5.3 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

##### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

##### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 692 of file basic\_test.c.

References dm7820\_ioctl\_region\_readwrite::access, dm7820\_ioctl\_region\_modify::access, access\_sizes, dm7820\_ioctl\_dma\_function::arguments, dm7820\_dma\_initialize\_arguments::buffer\_count, dm7820\_dma\_initialize\_arguments::buffer\_size, cleanup(), dm7820\_pci\_access\_request::data, dm7820\_pci\_access\_request::data16, dm7820\_pci\_access\_request::data32, dm7820\_pci\_access\_request::data8, device\_count, DM7820\_BAR0\_DMAPADR0, DM7820\_BAR0\_LENGTH, DM7820\_BAR2\_BOARD\_RESET, DM7820\_BAR2\_LENGTH, DM7820\_BAR2\_PORT0\_OUTPUT, DM7820\_BAR2\_PORT0\_PERIPH\_SEL\_L, DM7820\_BOARD\_RESET\_DO\_RESET, DM7820\_DMA\_FUNCTION\_INITIALIZE, DM7820\_DMA\_FUNCTION\_READ, DM7820\_DMA\_FUNCTION\_STOP, DM7820\_DMA\_FUNCTION\_WRITE, DM7820\_FIFO\_QUEUE\_0, DM7820\_FIFO\_QUEUE\_1, DM7820\_IOCTL\_DMA\_FUNCTION, DM7820\_IOCTL\_INTERRUPT\_STATUS, DM7820\_IOCTL\_REGION\_MODIFY, DM7820\_IOCTL\_REGION\_READ, DM7820\_IOCTL\_REGION\_WRITE, DM7820\_MAX\_DMA\_BUFFER\_COUNT, DM7820\_MAX\_DMA\_BUFFER\_SIZE, DM7820\_PCI\_REGION\_ACCESS\_16, DM7820\_PCI\_REGION\_ACCESS\_32, DM7820\_PCI\_REGION\_ACCESS\_8, DM7820\_PCI\_REGION\_FPGA\_MEM, DM7820\_PCI\_REGION\_PLX\_MEM, dm7820\_ioctl\_argument::dma\_function, dm7820\_dma\_function\_arguments::dma\_init, expect\_failure\_and\_check(), expect\_success(), register\_read::expected, dm7820\_ioctl\_dma\_function::fifo, dm7820\_ioctl\_dma\_function::function, INIT\_BAD\_DEVICE\_FILE\_CREATED, INIT\_BOARD\_ARRAY\_ALLOCATED, INIT\_BOARD\_A-

RRAY\_OPENED, dm7820\_ioctl\_argument::int\_status, dm7820\_ioctl\_region\_modify::mask, dm7820\_ioctl\_region\_modify::mask16, dm7820\_ioctl\_region\_modify::mask32, dm7820\_ioctl\_region\_modify::mask8, dm7820\_ioctl\_argument::modify, register\_read::offset, dm7820\_pci\_access\_request::offset, program\_name, dm7820\_ioctl\_argument::readwrite, register\_read::region, dm7820\_pci\_access\_request::region, region\_names, region\_sizes, register\_reads, register\_read::size, dm7820\_pci\_access\_request::size, dm7820\_ioctl\_dma\_function::transfer\_size, usage(), and dm7820\_ioctl\_interrupt\_status::wait\_for\_interrupt.

## 6.5.6 Variable Documentation

### 6.5.6.1 `uint8_t access_sizes[]` `[static]`

**Initial value:**

```
{
    8,
    16,
    32
}
```

Size in bits for each of the access sizes

Definition at line 175 of file basic\_test.c.

Referenced by main().

### 6.5.6.2 `volatile char bad_device_name[30]` `[static]`

Path name of DM7820 device file with invalid minor number

Definition at line 534 of file basic\_test.c.

### 6.5.6.3 `volatile int* descriptors` `[static]`

Array of DM7820 device file descriptors

Definition at line 528 of file basic\_test.c.

### 6.5.6.4 `volatile unsigned long device_count = 1` `[static]`

Number of devices found when driver was loaded

Definition at line 540 of file basic\_test.c.

Referenced by cleanup(), and main().

### 6.5.6.5 `volatile initialization_state_t init_state = INIT_NO_INITIALIZATION` `[static]`

Program initialization state

Definition at line 522 of file basic\_test.c.

**6.5.6.6** `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 169 of file `basic_test.c`.

Referenced by `main()`, and `usage()`.

**6.5.6.7** `char* region_names[]` `[static]`

**Initial value:**

```
{
    "BAR0",
    "BAR1",
    "BAR2"
}
```

PCI region names

Definition at line 225 of file `basic_test.c`.

Referenced by `main()`.

**6.5.6.8** `uint16_t region_sizes[]` `[static]`

**Initial value:**

```
{
    DM7820_BAR0_LENGTH,
    DM7820_BAR1_LENGTH,
    DM7820_BAR2_LENGTH
}
```

Length in bytes for each of the PCI regions

Definition at line 200 of file `basic_test.c`.

Referenced by `main()`.

**6.5.6.9** `register_read_t register_reads[]`

Data for register read tests. The board module ID registers are used because they have known values.

**Note**

These registers are 16 bits wide, therefore a 32-bit read of them must mask off the most significant 16 bits.

Definition at line 255 of file `basic_test.c`.

Referenced by `main()`.



## 6.6 examples/brdctl\_device\_info.c File Reference

Example program which demonstrates how to obtain version information and PCI master capable status from a device.

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

### Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* [program\\_name](#)

#### 6.6.1 Detailed Description

Example program which demonstrates how to obtain version information and PCI master capable status from a device.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

```
$Id: brdctl_device_info.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file [brdctl\\_device\\_info.c](#).

#### 6.6.2 Function Documentation

##### 6.6.2.1 int main ( int argument\_count, char \*\* arguments )

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

## Return values

<code>EXIT_SUCCESS</code>	Success.
<code>EXIT_FAILURE</code>	Failure.

Definition at line 95 of file `brdctl_device_info.c`.

References `board`, `DM7820_General_Close_Board()`, `DM7820_General_Get_Version_Info()`, `DM7820_General_Is_PCI_Master()`, `DM7820_General_Open_Board()`, `DM7820_Return_Status`, `program_name`, and `usage()`.

### 6.6.3 Variable Documentation

#### 6.6.3.1 `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 43 of file `brdctl_device_info.c`.

Referenced by `main()`, and `usage()`.

## 6.7 examples/dma\_capture\_buffers.c File Reference

Example program which demonstrates how to use DMA to continuously acquire samples.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "dm7820_library.h"
```

### Macros

- `#define BUF_SIZE 0x40000`
- `#define BUF_NUM 8`
- `#define SAMPLES ((BUF_SIZE * BUF_NUM)/2)`
- `#define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)`

### Functions

- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- void `ISR` (`dm7820_interrupt_info` `interrupt_info`)  
*Userspace ISR.*
- void `clean_up` ()  
*Disable and clean up.*
- int `main` (int `argument_count`, char `**arguments`)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)
- int [interrupts](#) = 0
- [DM7820\\_Board\\_Descriptor](#) \* [board](#)

### 6.7.1 Detailed Description

Example program which demonstrates how to use DMA to continuously acquire samples.

This example program demonstrates the DM7820's ability to continuously sample data from FIFO 0 by using DMA. This example will sample 8Mb of data (4 Million Samples) at a rate of 2.50 Mhz. We read DMA samples from the device then read them into a 8M buffer.

Warning: This example program is only intended to run on a system with enough RAM to support the large buffer's which will be allocated. Preferably 128MB minimum.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[dma\\_capture\\_buffers.c](#) 60276 2012-06-05 16:04:15Z rgroner

Definition in file [dma\\_capture\\_buffers.c](#).

### 6.7.2 Macro Definition Documentation

#### 6.7.2.1 #define BUF\_NUM 8

The number of buffers we want for each DMA channel.

Definition at line 89 of file [dma\\_capture\\_buffers.c](#).

Referenced by [main\(\)](#).

#### 6.7.2.2 #define BUF\_SIZE 0x40000

The size of the individual buffers we are asking to driver to create for each DMA channel. If you receive a memory error when trying to run this example, decrease this size.

Definition at line 83 of file [dma\\_capture\\_buffers.c](#).

Referenced by [main\(\)](#).

#### 6.7.2.3 #define DMA\_BUF\_SIZE (BUF\_SIZE \* BUF\_NUM)

The total space available in the device for DMA at any one time (4MB).

Definition at line 101 of file [dma\\_capture\\_buffers.c](#).

Referenced by [main\(\)](#).

#### 6.7.2.4 #define SAMPLES ((BUF\_SIZE \* BUF\_NUM)/2)

The number of samples that can be held by the list of buffers we created.

Definition at line 95 of file dma\_capture\_buffers.c.

### 6.7.3 Function Documentation

#### 6.7.3.1 void ISR ( dm7820\_interrupt\_info interrupt\_info )

Userspace ISR.

##### Parameters

<i>interrupt_info</i>	Information about the interrupt, returned by the kernel driver
-----------------------	--

Definition at line 144 of file dma\_capture\_buffers.c.

References DM7820\_INTERRUPT\_FIFO\_0\_DMA\_DONE, DM7820\_Return\_Status, \_dm7820\_interrupt\_info::error, interrupts, and \_dm7820\_interrupt\_info::source.

#### 6.7.3.2 int main ( int argument\_count, char \*\* arguments )

Main program code.

##### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

##### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 229 of file dma\_capture\_buffers.c.

References BUF\_NUM, BUF\_SIZE, clean\_up(), DM7820\_DMA\_DEMAND\_ON\_DM7820\_TO\_PCI, DM7820\_FIFO\_0\_DATA\_INPUT\_PORT\_0, DM7820\_FIFO\_DMA\_Configure(), DM7820\_FIFO\_DMA\_Create\_Buffer(), DM7820\_FIFO\_DMA\_Enable(), DM7820\_FIFO\_DMA\_Free\_Buffer(), DM7820\_FIFO\_DMA\_Initialize(), DM7820\_FIFO\_DMA\_Read(), DM7820\_FIFO\_DMA\_REQUEST\_READ, DM7820\_FIFO\_Enable(), DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0, DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_READ, DM7820\_FIFO\_QUEUE\_0, DM7820\_FIFO\_Set\_Data\_Input(), DM7820\_FIFO\_Set\_DMA\_Request(), DM7820\_FIFO\_Set\_Input\_Clock(), DM7820\_FIFO\_Set\_Output\_Clock(), DM7820\_General\_Enable\_Interrupt(), DM7820\_General\_Open\_Board(), DM7820\_General\_RemoveISR(), DM7820\_General\_Reset(), DM7820\_General\_SetISRPriority(), DM7820\_INTERRUPT\_FIFO\_0\_EMPTY, DM7820\_PRGCLK\_CLOCK\_0, DM7820\_PRGCLK\_MASTER\_25\_MHZ, DM7820\_PRGCLK\_MODE\_CONTINUOUS, DM7820\_PRGCLK\_MODE\_DISABLED, DM7820\_PrgClk\_Set\_Master(), DM7820\_PrgClk\_Set\_Mode(), DM7820\_PrgClk\_Set\_Period(), DM7820\_PrgClk\_Set\_Start\_Trigger(), DM7820\_PrgClk\_Set\_Stop\_Trigger(), DM7820\_PRGCLK\_START\_IMMEDIATE, DM7820\_PRGCLK\_STOP\_NONE, DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_PORT\_0, DM7820\_StdIO\_Set\_IO\_Mode(), DMA\_BUF\_SIZE, interrupts, ISR(), program\_name, and usage().

### 6.7.4 Variable Documentation

#### 6.7.4.1 DM7820\_Board\_Descriptor\* board

Device descriptor

Definition at line 75 of file dma\_capture\_buffers.c.

Referenced by main().

#### 6.7.4.2 int interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 69 of file dma\_capture\_buffers.c.

Referenced by ISR(), and main().

#### 6.7.4.3 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 63 of file dma\_capture\_buffers.c.

Referenced by main(), and usage().

## 6.8 examples/dma\_capture\_file.c File Reference

Example program which demonstrates how to use DMA to continuously acquire samples.

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "dm7820_library.h"
```

### Macros

- #define `DAT_FILE` `"/test.dat"`
- #define `BUF_SIZE` `0x40000`
- #define `BUF_NUM` `8`
- #define `SAMPLES` `((BUF_SIZE * BUF_NUM)/2)`
- #define `DMA_BUF_SIZE` `(BUF_SIZE * BUF_NUM)`

### Functions

- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- void `ISR` (`dm7820_interrupt_info` interrupt\_info)  
*Userspace ISR.*
- void `clean_up` ()  
*Disable and clean up.*

- int [main](#) (int argument\_count, char \*\*arguments)  
Main program code.

## Variables

- static char \* [program\\_name](#)
- volatile int [dma\\_done\\_interrupts](#) = 0
- [DM7820\\_Board\\_Descriptor](#) \* [board](#)
- int [test\\_data](#)

### 6.8.1 Detailed Description

Example program which demonstrates how to use DMA to continuously acquire samples.

This example program demonstrates the DM7820's ability to continuously sample data from FIFO 0 by using DMA. This example will sample 8Mb of data (4 Million Samples) at a rate of 1 Mhz. We read DMA samples from the device then dump them to a file on disk. We use FILE IO here as attempting to create buffers large enough to hold all the data we expect to gather can cause failures. Because we are using FILE I/O the speed at which we can continuously sample is restricted by how fast the I/O operations can finish.

Warning: This example program is only intended to run on a system with enough RAM to support the large buffer's which will be allocated. Preferably 128MB minimum.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[dma\\_capture\\_file.c](#) 79285 2014-05-21 20:37:42Z rgroner

Definition in file [dma\\_capture\\_file.c](#).

### 6.8.2 Macro Definition Documentation

#### 6.8.2.1 #define BUF\_NUM 8

The number of buffers we want for each DMA channel.

Definition at line 105 of file [dma\\_capture\\_file.c](#).

Referenced by [main\(\)](#).

#### 6.8.2.2 #define BUF\_SIZE 0x40000

The size of the individual buffers we are asking to driver to create for each DMA channel. If you receive a memory error when trying to run this example, decrease this size.

Definition at line 99 of file [dma\\_capture\\_file.c](#).

Referenced by [main\(\)](#).



FO\_Set\_Input\_Clock(), DM7820\_FIFO\_Set\_Output\_Clock(), DM7820\_FIFO\_STATUS\_EMPTY, DM7820\_General\_Enable\_Interrupt(), DM7820\_General\_Open\_Board(), DM7820\_General\_RemoveISR(), DM7820\_General\_Reset(), DM7820\_General\_SetISRPriority(), DM7820\_INTERRUPT\_FIFO\_0\_EMPTY, DM7820\_PRGCLK\_CLOCK\_0, DM7820\_PRGCLK\_MASTER\_25\_MHZ, DM7820\_PRGCLK\_MODE\_CONTINUOUS, DM7820\_PRGCLK\_MODE\_DISABLED, DM7820\_PrgClk\_Set\_Master(), DM7820\_PrgClk\_Set\_Mode(), DM7820\_PrgClk\_Set\_Period(), DM7820\_PrgClk\_Set\_Start\_Trigger(), DM7820\_PrgClk\_Set\_Stop\_Trigger(), DM7820\_PRGCLK\_START\_IMMEDIATE, DM7820\_PRGCLK\_STOP\_NONE, DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_PORT\_0, DM7820\_StdIO\_Set\_IO\_Mode(), DMA\_BUF\_SIZE, dma\_done\_interrupts, ISR(), program\_name, test\_data, and usage().

## 6.8.4 Variable Documentation

### 6.8.4.1 DM7820\_Board\_Descriptor\* board

Device descriptor

Definition at line 79 of file dma\_capture\_file.c.

### 6.8.4.2 volatile int dma\_done\_interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 73 of file dma\_capture\_file.c.

Referenced by ISR(), and main().

### 6.8.4.3 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 67 of file dma\_capture\_file.c.

Referenced by main(), and usage().

### 6.8.4.4 int test\_data

File handle for the data file.

Definition at line 85 of file dma\_capture\_file.c.

Referenced by main().

## 6.9 examples/fifo\_cascaded.c File Reference

Example program which demonstrates how to use the FIFO block FIFOs such that the output of one FIFO serves as the input of another FIFO.

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



## Functions

- static void `get_fifo_status` (`DM7820_Board_Descriptor *board`, `dm7820_fifo_queue fifo`, `dm7820_fifo_status-_condition condition`, `uint8_t *status`)  
*Determine whether or not the specified status condition has occurred for the given FIFO.*
- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int `main` (int `argument_count`, char `**arguments`)  
*Main program code.*

## Variables

- static char \* `program_name`

### 6.9.1 Detailed Description

Example program which demonstrates how to use the FIFO block FIFOs such that the output of one FIFO serves as the input of another FIFO.

This program transfers FIFO 0 data out standard I/O port 0, in standard I/O port 1, and into FIFO 1.

FIFO 0 is configured as follows: 1) input clock set to PCI write request, 2) output clock set to programmable clock 1, and 3) data input set to PCI data.

FIFO 1 is configured as follows: 1) input clock set to programmable clock 1, 2) output clock set to PCI read request, and 3) data input set to standard I/O port 1.

The programmable clocks are set up as follows:

Clock	Master Clock	Start Trigger	Stop Trigger	Mode	Frequency
0	25 MHz	immediate	none	continuous	100 KHz
1	Clock 0	immediate	none	continuous	2 Hz

Standard I/O ports 0 and 1 should be connected as follows:

CN10 Pin	CN11 Pin
17	17
19	19
21	21
23	23
25	25
27	27
29	29
31	31
33	33
35	35
37	37
39	39
41	41
43	43
45	45
47	47

A character string is written to FIFO 0. Once the programmable clocks are started, the string is transferred to FIFO 1 at the rate of two characters per second. While the characters are being transferred, the program reads the characters from FIFO 1 and reassembles the string.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[fifo\\_cascaded.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [fifo\\_cascaded.c](#).

## 6.9.2 Function Documentation

**6.9.2.1** `static void get_fifo_status ( DM7820_Board_Descriptor * board, dm7820_fifo_queue fifo, dm7820_fifo_status_condition condition, uint8_t * status ) [static]`

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

### Parameters

<i>board</i>	Address of device's library board descriptor.
<i>fifo</i>	The FIFO to determine status of.
<i>condition</i>	The status condition to check for.
<i>status</i>	Address where occurrence flag should be stored.

Definition at line 126 of file [fifo\\_cascaded.c](#).

References [DM7820\\_FIFO\\_Get\\_Status\(\)](#).

Referenced by [main\(\)](#).

**6.9.2.2** `int main ( int argument_count, char ** arguments )`

Main program code.

### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 186 of file [fifo\\_cascaded.c](#).

References [board](#), [DM7820\\_FIFO\\_0\\_DATA\\_INPUT\\_PCI\\_DATA](#), [DM7820\\_FIFO\\_1\\_DATA\\_INPUT\\_PORT\\_1](#), [DM7820\\_FIFO\\_Enable\(\)](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PCI\\_WRITE](#), [DM7820\\_FIFO\\_INPUT\\_CLOCK\\_PROG\\_CLOCK\\_1](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_PCI\\_READ](#), [DM7820\\_FIFO\\_OUTPUT\\_CLOCK\\_PROG\\_CLOCK\\_1](#), [DM7820\\_FIFO\\_QUEUE\\_0](#), [DM7820\\_FIFO\\_QUEUE\\_1](#), [DM7820\\_FIFO\\_Read\(\)](#), [DM7820\\_FIFO\\_Set\\_Data\\_Input\(\)](#), [DM7820\\_FIFO\\_Set\\_Input\\_Clock\(\)](#), [DM7820\\_FIFO\\_Set\\_Output\\_Clock\(\)](#), [DM7820\\_FIFO\\_STATUS\\_EMPTY](#), [DM7820\\_FIFO\\_STATUS\\_FULL](#), [DM7820\\_FIFO\\_STATUS\\_OVERFLOW](#), [DM7820\\_FIFO\\_STATUS\\_UNDERFLOW](#), [DM7820\\_FIFO\\_Write\(\)](#), [DM7820\\_General\\_Close\\_Board\(\)](#), [DM7820\\_General\\_Open\\_Board\(\)](#), [DM7820\\_PRGCLK\\_CLOCK\\_0](#), [DM7820\\_PRGCLK\\_CLOCK\\_1](#), [DM7820\\_PRGCLK\\_CLOCK\\_2](#), [DM7820\\_PRGCLK\\_CLOCK\\_3](#),

DM7820\_PRGCLK\_MASTER\_25\_MHZ, DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_0, DM7820\_PRGCLK\_MODE\_CONTINUOUS, DM7820\_PRGCLK\_MODE\_DISABLED, DM7820\_PrgClk\_Set\_Master(), DM7820\_PrgClk\_Set\_Mode(), DM7820\_PrgClk\_Set\_Period(), DM7820\_PrgClk\_Set\_Start\_Trigger(), DM7820\_PrgClk\_Set\_Stop\_Trigger(), DM7820\_PRGCLK\_START\_IMMEDIATE, DM7820\_PRGCLK\_STOP\_NONE, DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_MODE\_PER\_OUT, DM7820\_STDIO\_PERIPH\_FIFO\_0, DM7820\_STDIO\_PORT\_0, DM7820\_STDIO\_PORT\_1, DM7820\_StdIO\_Set\_IO\_Mode(), DM7820\_StdIO\_Set\_Periph\_Mode(), get\_fifo\_status(), program\_name, and usage().

### 6.9.3 Variable Documentation

#### 6.9.3.1 `char* program_name` [static]

Name of the program as invoked on the command line

Definition at line 91 of file fifo\_cascaded.c.

Referenced by main(), and usage().

## 6.10 examples/fifo\_interrupt.c File Reference

Example program which demonstrates a loopback accross the FIFO's using DMA.

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

### Functions

- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* `program_name`

#### 6.10.1 Detailed Description

Example program which demonstrates a loopback accross the FIFO's using DMA.

This program uses FIFO 0, which is configured as follows: 1) input clock set to PCI write request, 2) output clock set to programmable clock 1, and 3) data input set to PCI data.

The programmable clocks are set up as follows:

Master	Start	Stop
--------	-------	------

Clock	Clock	Trigger	Trigger	Mode	Frequency
0	25 MHz	immediate	none	continuous	500 Hz
1	Clock 0	immediate	none	continuous	1 Hz

With the setup indicated above, programmable clock 1 will cause a value to be read from FIFO 0 once every second.

Any value clocked out of FIFO 0 is discarded.

The program loops performing the following sequence of actions:

1) writing 5 values to FIFO 0, 2) enabling FIFO 0 empty interrupt, 3) starting programmable clocks 0 and 1, 4) waiting for a FIFO 0 empty interrupt, and 5) disabling programmable clocks 0 and 1.

The above sequence of actions is performed 4 times and then the program exits.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[fifo\\_interrupt.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [fifo\\_interrupt.c](#).

## 6.10.2 Function Documentation

### 6.10.2.1 `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 134 of file `fifo_interrupt.c`.

References `board`, `DM7820_FIFO_0_DATA_INPUT_PCI_DATA`, `DM7820_FIFO_Enable()`, `DM7820_FIFO_INPUT_CLOCK_PCI_WRITE`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1`, `DM7820_FIFO_QUEUE_0`, `DM7820_FIFO_Set_Data_Input()`, `DM7820_FIFO_Set_Input_Clock()`, `DM7820_FIFO_Set_Output_Clock()`, `DM7820_FIFO_Write()`, `DM7820_General_Close_Board()`, `DM7820_General_Enable_Interrupt()`, `DM7820_General_Get_Interrupt_Status()`, `DM7820_General_Open_Board()`, `DM7820_General_Reset()`, `DM7820_INTERRUPT_FIFO_0_EMPTY`, `DM7820_PRGCLK_CLOCK_0`, `DM7820_PRGCLK_CLOCK_1`, `DM7820_PRGCLK_MASTER_25_MHZ`, `DM7820_PRGCLK_MASTER_PROG_CLOCK_0`, `DM7820_PRGCLK_MODE_CONTINUOUS`, `DM7820_PRGCLK_MODE_DISABLED`, `DM7820_PrgClk_Set_Master()`, `DM7820_PrgClk_Set_Mode()`, `DM7820_PrgClk_Set_Period()`, `DM7820_PrgClk_Set_Start_Trigger()`, `DM7820_PrgClk_Set_Stop_Trigger()`, `DM7820_PRGCLK_START_IMMEDIATE`, `DM7820_PRGCLK_STOP_NONE`, `DM7820_Return_Status`, `program_name`, `_dm7820_interrupt_info::source`, and `usage()`.

### 6.10.3 Variable Documentation

#### 6.10.3.1 `char* program_name` [static]

Name of the program as invoked on the command line

Definition at line 72 of file `fifo_interrupt.c`.

Referenced by `main()`, and `usage()`.

## 6.11 examples/fifo\_pci\_access.c File Reference

Example program which demonstrates how to use the FIFO block FIFOs with PCI read requests clocking data out of the FIFO and PCI write requests clocking data into the FIFO.

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

### Macros

- `#define SDRAM_FIFO_ELEMENTS`  $((1024 * 1024 * 2) - 1)$
- `#define INPUT_FIFO_ELEMENTS`  $(256 - 1)$
- `#define OUTPUT_FIFO_ELEMENTS`  $(256 - 1)$
- `#define EXPECTED_FIFO_ELEMENTS`
- `#define MODULUS_DIVISOR` 4321

### Functions

- static void `get_fifo_status` (`DM7820_Board_Descriptor` \*board, `dm7820_fifo_queue` fifo, `dm7820_fifo_status_`  
`_condition` condition, `uint8_t` \*status)  
*Determine whether or not the specified status condition has occurred for the given FIFO.*
- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* `program_name`

#### 6.11.1 Detailed Description

Example program which demonstrates how to use the FIFO block FIFOs with PCI read requests clocking data out of the FIFO and PCI write requests clocking data into the FIFO.

**Note**

This program shows how to access FIFOs at the most basic level.  
Because this program does not use interrupts, it also demonstrates how to check FIFO status when interrupts are not used.

**Warning**

This program takes a long time to complete.

```
This program uses FIFO 0, which is configured as follows: 1) input clock
set to PCI write request, 2) output clock set to PCI read request, and
3) data input set to PCI data.
```

```
Values are written to FIFO 0 until the FIFO is full. Then, values are
read from FIFO 0 until the FIFO is empty. While reading values, a check
is made to see that the values are read back in the order they were
written.
```

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

```
$Id: fifo_pci_access.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file [fifo\\_pci\\_access.c](#).

**6.11.2 Macro Definition Documentation****6.11.2.1 #define EXPECTED\_FIFO\_ELEMENTS****Value:**

```
( \
    SDRAM_FIFO_ELEMENTS + INPUT_FIFO_ELEMENTS +
    OUTPUT_FIFO_ELEMENTS \
)
```

Expected number of 16-bit FIFO elements

Definition at line 86 of file [fifo\\_pci\\_access.c](#).

Referenced by [main\(\)](#).

**6.11.2.2 #define INPUT\_FIFO\_ELEMENTS (256 - 1)**

256 16-bit elements in input FIFO; one element is lost for hardware to determine full status

Definition at line 73 of file [fifo\\_pci\\_access.c](#).

**6.11.2.3 #define MODULUS\_DIVISOR 4321**

Divisor for modulus operator; used when writing data to FIFO that can be easily recalculated when reading data from FIFO

Definition at line 95 of file [fifo\\_pci\\_access.c](#).

Referenced by [main\(\)](#).

6.11.2.4 `#define OUTPUT_FIFO_ELEMENTS (256 - 1)`

256 16-bit elements in output FIFO; one element is lost for hardware to determine full status

Definition at line 80 of file `fifo_pci_access.c`.

6.11.2.5 `#define SDRAM_FIFO_ELEMENTS ((1024 * 1024 * 2) - 1)`

2 megabyte 16-bit elements in SDRAM FIFO; one element is lost for hardware to determine full status

Definition at line 65 of file `fifo_pci_access.c`.

## 6.11.3 Function Documentation

6.11.3.1 `static void get_fifo_status ( DM7820_Board_Descriptor * board, dm7820_fifo_queue fifo, dm7820_fifo_status_condition condition, uint8_t * status ) [static]`

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

## Parameters

<i>board</i>	Address of device's library board descriptor.
<i>fifo</i>	The FIFO to determine status of.
<i>condition</i>	The status condition to check for.
<i>status</i>	Address where occurrence flag should be stored.

Definition at line 140 of file `fifo_pci_access.c`.

References `DM7820_FIFO_Get_Status()`.

Referenced by `main()`.

6.11.3.2 `int main ( int argument_count, char ** arguments )`

Main program code.

## Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

## Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 200 of file `fifo_pci_access.c`.

References `board`, `DM7820_FIFO_0_DATA_INPUT_PCI_DATA`, `DM7820_FIFO_Enable()`, `DM7820_FIFO_INPUT_CLOCK_PCI_WRITE`, `DM7820_FIFO_OUTPUT_CLOCK_PCI_READ`, `DM7820_FIFO_QUEUE_0`, `DM7820_FIFO_QUEUE_1`, `DM7820_FIFO_Read()`, `DM7820_FIFO_Set_Data_Input()`, `DM7820_FIFO_Set_Input_Clock()`, `DM7820_FIFO_Set_Output_Clock()`, `DM7820_FIFO_STATUS_EMPTY`, `DM7820_FIFO_STATUS_FULL`, `DM7820_FIFO_STATUS_OVERFLOW`, `DM7820_FIFO_STATUS_UNDERFLOW`, `DM7820_FIFO_Write()`, `DM7820_General_Close_Board()`, `DM7820_General_Open_Board()`, `DM7820_Return_Status`, `EXPECTED_FIFO_ELEMENTS`, `get_fifo_status()`, `MODULUS_DIVISOR`, `program_name`, and `usage()`.

## 6.11.4 Variable Documentation

#### 6.11.4.1 `char* program_name` [static]

Name of the program as invoked on the command line

Definition at line 105 of file `fifo_pci_access.c`.

Referenced by `main()`, and `usage()`.

## 6.12 `examples/incenc_encoders.c` File Reference

Example program which demonstrates how to use the incremental encoder block encoders.

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

### Functions

- static void `sigint_handler` (int signal\_number)  
*Signal handler for SIGINT Control-C keyboard interrupt.*
- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* `program_name`
- static volatile sig\_atomic\_t `exit_program` = 0

#### 6.12.1 Detailed Description

Example program which demonstrates how to use the incremental encoder block encoders.

Incremental encoders 0 and 1 are configured as follows: 1) master clock set to 25 MHz clock, 2) value register hold disabled, 3) single-ended inputs, 4) input filter disabled, 5) independent A/B channels, and 6) index input disabled.

Incremental encoder 0 is further configured with an initial value of 0xFFFF and a phase filter to disable counter change on up transitions.

Incremental encoder 1 is further configured with an initial value of 0x0000 and a phase filter to disable counter change on down transitions.

This program assumes a single incremental encoder is connected to incremental encoder 0 channel A and to incremental encoder 1 channel A. The incremental encoder should be connected as follows:

Encoder Pin	CN11 Pin	CN10 Pin
-------------	----------	----------



```
=====
A          47          47
B          43          43
-          N/A         50
+          N/A         49
```

With the setup indicated above, encoder 0 channel A counts down only when the incremental encoder is rotated in one direction and encoder 1 channel A counts up only when the incremental encoder is rotated in the opposite direction.

Channel B on incremental encoders 0 and 1 is not used.

#### Note

Because all four phase filter transitions are enabled in a particular counting direction (up or down), the counter value will update at four times the incremental encoder "cycles per revolution" rate.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[incenc\\_encoders.c](#) 79285 2014-05-21 20:37:42Z rgroner

Definition in file [incenc\\_encoders.c](#).

## 6.12.2 Function Documentation

### 6.12.2.1 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 179 of file incenc\_encoders.c.

References board, DM7820\_General\_Close\_Board(), DM7820\_General\_Open\_Board(), DM7820\_INCENC\_CHANNEL\_A, DM7820\_INCENC\_CHANNEL\_INDEPENDENT, DM7820\_IncEnc\_Configure(), DM7820\_INCENC\_DISABLE\_PHASE\_FILTER\_TRANSITION, DM7820\_IncEnc\_Enable(), DM7820\_IncEnc\_Enable\_Hold(), DM7820\_INCENC\_ENCODER\_0, DM7820\_INCENC\_ENCODER\_1, DM7820\_IncEnc\_Get\_Independent\_Value(), DM7820\_INCENC\_INPUT\_SINGLE\_ENDED, DM7820\_INCENC\_MASTER\_25\_MHZ, DM7820\_INCENC\_PHASE\_BA\_00\_TO\_01\_UP, DM7820\_INCENC\_PHASE\_BA\_00\_TO\_10\_DOWN, DM7820\_INCENC\_PHASE\_BA\_01\_TO\_00\_DOWN, DM7820\_INCENC\_PHASE\_BA\_01\_TO\_11\_UP, DM7820\_INCENC\_PHASE\_BA\_10\_TO\_00\_UP, DM7820\_INCENC\_PHASE\_BA\_10\_TO\_11\_DOWN, DM7820\_INCENC\_PHASE\_BA\_11\_TO\_01\_DOWN, DM7820\_INCENC\_PHASE\_BA\_11\_TO\_10\_UP, DM7820\_INCENC\_RESET\_PHASE\_FILTER, DM7820\_IncEnc\_Set\_Independent\_Value(), DM7820\_IncEnc\_Set\_Master(), DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_INPUT, DM7820\_ST-

DIO\_PORT\_0, DM7820\_STDIO\_PORT\_1, DM7820\_StdIO\_Set\_IO\_Mode(), exit\_program, program\_name, sigint\_handler(), and usage().

#### 6.12.2.2 static void sigint\_handler ( int *signal\_number* ) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

##### Parameters

<i>signal_number</i>	Signal number passed in from kernel.
----------------------	--------------------------------------

##### Warning

One must be extremely careful about what functions are called from a signal handler. printf() and related functions are considered unsafe for use in signal handlers. Therefore, this function uses write() instead.

Definition at line 109 of file incenc\_encoders.c.

References exit\_program.

Referenced by main().

### 6.12.3 Variable Documentation

#### 6.12.3.1 volatile sig\_atomic\_t exit\_program = 0 [static]

Flag used by SIGINT signal handler to tell main program to exit because the user hit Control-C

Definition at line 86 of file incenc\_encoders.c.

Referenced by main(), and sigint\_handler().

#### 6.12.3.2 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 79 of file incenc\_encoders.c.

Referenced by main(), and usage().

## 6.13 examples/incenc\_interrupt.c File Reference

Example program which demonstrates how to use the incremental encoder block interrupts.

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

## Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)

### 6.13.1 Detailed Description

Example program which demonstrates how to use the incremental encoder block interrupts.

Incremental encoder 0 is configured as follows: 1) master clock set to 25 MHz clock, 2) value register hold disabled, 3) single-ended inputs, 4) input filter disabled, 5) independent A/B channels, and 6) index input disabled.

Incremental encoder 0 is further configured with an initial value of 0x8000 and a phase filter to enable counter change on all transitions.

This program assumes an incremental encoder is connected to incremental encoder 0 channel A. The incremental encoder should be connected as follows:

Encoder Pin	CN10 Pin
=====	=====
A	47
B	43
-	50
+	49

Channel B on incremental encoder 0 is not used.

This program uses incremental encoder 0 channel A negative & positive rollover interrupts and waits for the interrupts to occur. One such interrupt is waited on and then the program exits.

## Note

Because all phase filter transitions are enabled in a particular counting direction (up or down), the counter value will update at four times the incremental encoder "cycles per revolution" rate.

This program does not display the channel A value because sleepy waiting may be used.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

## Id:

[incenc\\_interrupt.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [incenc\\_interrupt.c](#).

## 6.13.2 Function Documentation

### 6.13.2.1 `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 142 of file `incenc_interrupt.c`.

References `board`, `DM7820_General_Close_Board()`, `DM7820_General_Enable_Interrupt()`, `DM7820_General_Get_Interrupt_Status()`, `DM7820_General_Open_Board()`, `DM7820_General_Reset()`, `DM7820_INCENC_CHANNEL_A`, `DM7820_INCENC_CHANNEL_INDEPENDENT`, `DM7820_IncEnc_Configure()`, `DM7820_IncEnc_Enable()`, `DM7820_IncEnc_Enable_Hold()`, `DM7820_INCENC_ENCODER_0`, `DM7820_INCENC_INPUT_SINGLE_ENDED`, `DM7820_INCENC_MASTER_25_MHZ`, `DM7820_INCENC_RESET_PHASE_FILTER`, `DM7820_IncEnc_Set_Independent_Value()`, `DM7820_IncEnc_Set_Master()`, `DM7820_INTERRUPT_INCENC_0_CHANNEL_A_NEGATIVE_ROLLOVER`, `DM7820_INTERRUPT_INCENC_0_CHANNEL_A_POSITIVE_ROLLOVER`, `DM7820_Return_Status`, `DM7820_STDIO_MODE_INPUT`, `DM7820_STDIO_PORT_0`, `DM7820_StdIO_Set_IO_Mode()`, `program_name`, `_dm7820_interrupt_info::source`, and `usage()`.

## 6.13.3 Variable Documentation

### 6.13.3.1 `char* program_name [static]`

Name of the program as invoked on the command line

Definition at line 79 of file `incenc_interrupt.c`.

Referenced by `main()`, and `usage()`.

## 6.14 `examples/incenc_status.c` File Reference

Example program which demonstrates how to get incremental encoder status when not using interrupts.

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

### Functions

- static void `usage` (void)

*Print information on stderr about how the program is to be used. After doing so, the program is exited.*

- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)

### 6.14.1 Detailed Description

Example program which demonstrates how to get incremental encoder status when not using interrupts.

Incremental encoder 0 is configured as follows: 1) master clock set to 25 MHz clock, 2) value register hold disabled, 3) single-ended inputs, 4) input filter disabled, 5) independent A/B channels, and 6) index input disabled.

Incremental encoder 0 is further configured with an initial value of 0x8000 and a phase filter to enable counter change on all transitions.

This program assumes an incremental encoder is connected to incremental encoder 0 channel A. The incremental encoder should be connected as follows:

Encoder Pin	CN10 Pin
=====	
A	47
B	43
-	50
+	49

Encoder 0 channel A values are read and printed until either negative rollover or positive rollover occurs.

Channel B on incremental encoder 0 is not used.

## Note

Because all phase filter transitions are enabled in a particular counting direction (up or down), the counter value will update at four times the incremental encoder "cycles per revolution" rate.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[incenc\\_status.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [incenc\\_status.c](#).

### 6.14.2 Function Documentation

#### 6.14.2.1 int main ( int argument\_count, char \*\* arguments )

Main program code.

**Parameters**

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

**Return values**

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 129 of file incenc\_status.c.

References board, DM7820\_General\_Close\_Board(), DM7820\_General\_Open\_Board(), DM7820\_INCENC\_CHANNEL\_A, DM7820\_INCENC\_CHANNEL\_INDEPENDENT, DM7820\_IncEnc\_Configure(), DM7820\_IncEnc\_Enable(), DM7820\_IncEnc\_Enable\_Hold(), DM7820\_INCENC\_ENCODER\_0, DM7820\_IncEnc\_Get\_Independent\_Value(), DM7820\_IncEnc\_Get\_Status(), DM7820\_INCENC\_INPUT\_SINGLE\_ENDED, DM7820\_INCENC\_MASTER\_25\_MHZ, DM7820\_INCENC\_RESET\_PHASE\_FILTER, DM7820\_IncEnc\_Set\_Independent\_Value(), DM7820\_IncEnc\_Set\_Master(), DM7820\_INCENC\_STATUS\_CHANNEL\_A\_NEGATIVE\_ROLLOVER, DM7820\_INCENC\_STATUS\_CHANNEL\_A\_POSITIVE\_ROLLOVER, DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_PORT\_0, DM7820\_StdIO\_Set\_IO\_Mode(), program\_name, and usage().

**6.14.3 Variable Documentation****6.14.3.1 char\* program\_name [static]**

Name of the program as invoked on the command line

Definition at line 73 of file incenc\_status.c.

Referenced by main(), and usage().

**6.15 examples/library\_test.c File Reference**

Program which tests the basic functionality of the user library.

```
#include <errno.h>
#include <error.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/utsname.h>
#include <unistd.h>
#include "dm7820_ioctl.h"
#include "dm7820_library.h"
#include "dm7820_registers.h"
```

## Macros

- `#define DM7820_MAX_DMA_BUFFER_COUNT 32`
- `#define DM7820_MAX_DMA_BUFFER_SIZE 0x20000`

## Typedefs

- `typedef enum initialization_state initialization_state_t`

## Enumerations

- `enum initialization_state {`  
`INIT_NO_INITIALIZATION = 0x00000000, INIT_BOARD_ARRAY_ALLOCATED = 0x00000004, INIT_BAD-`  
`_DEVICE_FILE_CREATED = 0x00000008, INIT_BOARD_ARRAY_OPENED = 0x00000010,`  
`INIT_NO_INITIALIZATION = 0x00000000, INIT_BOARD_ARRAY_ALLOCATED = 0x00000004, INIT_BAD-`  
`_DEVICE_FILE_CREATED = 0x00000008, INIT_BOARD_ARRAY_OPENED = 0x00000010 }`

## Functions

- `static void cleanup (void)`  
*Perform actions necessary to clean up after the program encounters a fatal error.*
- `static void expect_failure_and_check (DM7820_Error status, int expected_errno)`  
*Verify that a function being tested failed with the given errno.*
- `static void expect_success (DM7820_Error status)`  
*Verify that a function being tested succeeded.*
- `static void usage (void)`  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- `int main (int argument_count, char **arguments)`  
*Main program code.*

## Variables

- `static char * program_name`
- `static volatile initialization_state_t init_state = INIT_NO_INITIALIZATION`
- `static volatile DM7820_Board_Descriptor ** boards`
- `static volatile char bad_device_name [30]`
- `static volatile unsigned long device_count = 1`

### 6.15.1 Detailed Description

Program which tests the basic functionality of the user library.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[library\\_test.c](#) 86306 2015-03-05 15:27:53Z rgroner

Definition in file [library\\_test.c](#).

## 6.15.2 Macro Definition Documentation

### 6.15.2.1 `#define DM7820_MAX_DMA_BUFFER_COUNT 32`

Default driver limit on number of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 52 of file [library\\_test.c](#).

Referenced by [main\(\)](#).

### 6.15.2.2 `#define DM7820_MAX_DMA_BUFFER_SIZE 0x20000`

Default driver limit in bytes on size of DMA buffers that can be allocated per DMA/FIFO channel

Definition at line 59 of file [library\\_test.c](#).

Referenced by [main\(\)](#).

## 6.15.3 Typedef Documentation

### 6.15.3.1 `typedef enum initialization_state initialization_state_t`

Program initialization status type

Definition at line 106 of file [library\\_test.c](#).

## 6.15.4 Enumeration Type Documentation

### 6.15.4.1 `enum initialization_state`

Status of program initialization. Used to indication which operations need to be undone when cleaning up after an error. This is a bit mask, so only one bit should be set for each enumeration value.

Enumerator:

**`INIT_NO_INITIALIZATION`** No initialization performed

**`INIT_BOARD_ARRAY_ALLOCATED`** File descriptor array allocated

**`INIT_BAD_DEVICE_FILE_CREATED`** /dev entry with invalid minor number created

**`INIT_BOARD_ARRAY_OPENED`** File descriptor array contains at least one opened file

**`INIT_NO_INITIALIZATION`** No initialization performed

**`INIT_BOARD_ARRAY_ALLOCATED`** Board descriptor array allocated

**`INIT_BAD_DEVICE_FILE_CREATED`** /dev entry with invalid minor number created

**`INIT_BOARD_ARRAY_OPENED`** Board descriptor array contains at least one opened file

Definition at line 71 of file [library\\_test.c](#).



### 6.15.5 Function Documentation

**6.15.5.1** `static void expect_failure_and_check ( DM7820_Error status, int expected_errno )` `[static]`

Verify that a function being tested failed with the given errno.

#### Parameters

<i>status</i>	Return code from function being tested.
<i>expected_errno</i>	Expected errno that function should have set.

Definition at line 190 of file library\_test.c.

References `cleanup()`.

Referenced by `main()`.

**6.15.5.2** `static void expect_success ( DM7820_Error status )` `[static]`

Verify that a function being tested succeeded.

#### Parameters

<i>status</i>	Return code from function being tested.
---------------	---

Definition at line 218 of file library\_test.c.

References `cleanup()`.

Referenced by `main()`.

**6.15.5.3** `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 276 of file library\_test.c.

References `dm7820_ioctl_region_readwrite::access`, `board`, `cleanup()`, `dm7820_pci_access_request::data`, `dm7820_pci_access_request::data16`, `device_count`, `DM7820_AdvInt_Get_Status()`, `DM7820_ADVINT_INTERRUPT_0`, `DM7820_ADVINT_INTERRUPT_1`, `DM7820_ADVINT_MASTER_25_MHZ`, `DM7820_ADVINT_MASTER_INV_STROBE_2`, `DM7820_ADVINT_MASTER_RESERVED`, `DM7820_ADVINT_MODE_DISABLED`, `DM7820_ADVINT_MODE_EVENT`, `DM7820_AdvInt_Read_Capture()`, `DM7820_AdvInt_Set_Compare()`, `DM7820_AdvInt_Set_Mask()`, `DM7820_AdvInt_Set_Master()`, `DM7820_AdvInt_Set_Mode()`, `DM7820_BAR2_BRD_STAT`, `DM7820_BAR2_PORT0_OUTPUT`, `DM7820_DMA_DEMAND_ON_PCI_TO_DM7820`, `DM7820_FIFO_0_DATA_INPUT_PCI_DATA`, `DM7820_FIFO_1_DATA_INPUT_INC_ENCODER_1_B`, `DM7820_FIFO_1_DATA_INPUT_PCI_DATA`, `DM7820_FIFO_DMA_Configure()`, `DM7820_FIFO_DMA_Initialize()`, `DM7820_FIFO_DMA_REQUEST_NOT_FULL`, `DM7820_FIFO_DMA_REQUEST_READ`, `DM7820_FIFO_Enable()`, `DM7820_FIFO_Get_Status()`, `DM7820_FIFO_INPUT_CLOCK_25_MHZ`, `DM7820_FIFO_INPUT_CLOCK_PCI_WRITE`, `DM7820_FIFO_INPUT_CLOCK_RESERVED_1`, `DM7820_FIFO_INPUT_CLOCK_RESERVED_2`, `DM7820_FIFO_INPUT_CLOCK_`

RESERVED\_3, DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_4, DM7820\_FIFO\_OUTPUT\_CLOCK\_25\_MHZ, DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_WRITE, DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_1, DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_2, DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_3, DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_4, DM7820\_FIFO\_QUEUE\_0, DM7820\_FIFO\_QUEUE\_1, DM7820\_FIFO\_Read(), DM7820\_FIFO\_Set\_Data\_Input(), DM7820\_FIFO\_Set\_DMA\_Request(), DM7820\_FIFO\_Set\_Input\_Clock(), DM7820\_FIFO\_Set\_Output\_Clock(), DM7820\_FIFO\_STATUS\_EMPTY, DM7820\_FIFO\_STATUS\_READ\_REQUEST, DM7820\_FIFO\_STATUS\_UNDERFLOW, DM7820\_FIFO\_Write(), DM7820\_General\_Close\_Board(), DM7820\_General\_Enable\_Interrupt(), DM7820\_General\_Get\_Interrupt\_Status(), DM7820\_General\_Get\_Version\_Info(), DM7820\_General\_Is\_PCI\_Master(), DM7820\_General\_Open\_Board(), DM7820\_General\_Reset(), DM7820\_INCENC\_CHANNEL\_A, DM7820\_INCENC\_CHANNEL\_B, DM7820\_INCENC\_CHANNEL\_INDEPENDENT, DM7820\_INCENC\_CHANNEL\_JOINED, DM7820\_IncEnc\_Configure(), DM7820\_IncEnc\_Enable(), DM7820\_IncEnc\_Enable\_Hold(), DM7820\_INCENC\_ENCODER\_0, DM7820\_INCENC\_ENCODER\_1, DM7820\_IncEnc\_Get\_Independent\_Value(), DM7820\_IncEnc\_Get\_Joined\_Value(), DM7820\_IncEnc\_Get\_Status(), DM7820\_INCENC\_INPUT\_DIFFERENTIAL, DM7820\_INCENC\_INPUT\_SINGLE\_ENDED, DM7820\_INCENC\_MASTER\_25\_MHZ, DM7820\_INCENC\_MASTER\_INV\_STROBE\_2, DM7820\_INCENC\_MASTER\_RESERVED, DM7820\_IncEnc\_Set\_Independent\_Value(), DM7820\_IncEnc\_Set\_Joined\_Value(), DM7820\_IncEnc\_Set\_Master(), DM7820\_INCENC\_STATUS\_CHANNEL\_A\_POSITIVE\_ROLLOVER, DM7820\_INCENC\_STATUS\_CHANNEL\_B\_NEGATIVE\_ROLLOVER, DM7820\_INTERRUPT\_ADVINT\_0, DM7820\_INTERRUPT\_FIFO\_0\_DMA\_DONE, DM7820\_INTERRUPT\_FIFO\_1\_DMA\_DONE, DM7820\_IOCTL\_REGION\_READ, DM7820\_IOCTL\_REGION\_WRITE, DM7820\_MAX\_DMA\_BUFFER\_COUNT, DM7820\_MAX\_DMA\_BUFFER\_SIZE, DM7820\_PCI\_REGION\_ACCESS\_16, DM7820\_PCI\_REGION\_FPGA\_MEM, DM7820\_PRGCLK\_CLOCK\_0, DM7820\_PRGCLK\_CLOCK\_3, DM7820\_PRGCLK\_MASTER\_25\_MHZ, DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_2, DM7820\_PRGCLK\_MODE\_CONTINUOUS, DM7820\_PRGCLK\_MODE\_DISABLED, DM7820\_PRGCLK\_MODE\_ONE\_SHOT, DM7820\_PRGCLK\_MODE\_RESERVED, DM7820\_PrgClk\_Set\_Master(), DM7820\_PrgClk\_Set\_Mode(), DM7820\_PrgClk\_Set\_Period(), DM7820\_PrgClk\_Set\_Start\_Trigger(), DM7820\_PrgClk\_Set\_Stop\_Trigger(), DM7820\_PRGCLK\_START\_FIFO\_1\_INT, DM7820\_PRGCLK\_START\_IMMEDIATE, DM7820\_PRGCLK\_START\_RESERVED\_1, DM7820\_PRGCLK\_START\_RESERVED\_2, DM7820\_PRGCLK\_START\_RESERVED\_3, DM7820\_PRGCLK\_START\_RESERVED\_4, DM7820\_PRGCLK\_STOP\_FIFO\_1\_INT, DM7820\_PRGCLK\_STOP\_NONE, DM7820\_PRGCLK\_STOP\_RESERVED\_1, DM7820\_PRGCLK\_STOP\_RESERVED\_2, DM7820\_PRGCLK\_STOP\_RESERVED\_3, DM7820\_PRGCLK\_STOP\_RESERVED\_4, DM7820\_PWM\_Enable(), DM7820\_PWM\_OUTPUT\_A, DM7820\_PWM\_OUTPUT\_D, DM7820\_PWM\_PERIOD\_MASTER\_25\_MHZ, DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_2, DM7820\_PWM\_PERIOD\_MASTER\_RESERVED, DM7820\_PWM\_Set\_Period(), DM7820\_PWM\_Set\_Period\_Master(), DM7820\_PWM\_Set\_Width(), DM7820\_PWM\_Set\_Width\_Master(), DM7820\_PWM\_WIDTH\_MASTER\_25\_MHZ, DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_2, DM7820\_PWM\_WIDTH\_MASTER\_RESERVED, DM7820\_StdIO\_Get\_Input(), DM7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_MODE\_PER\_OUT, DM7820\_STDIO\_PERIPH\_CLK\_OTHER, DM7820\_STDIO\_PERIPH\_FIFO\_0, DM7820\_STDIO\_PERIPH\_FIFO\_1, DM7820\_STDIO\_PERIPH\_PWM, DM7820\_STDIO\_PORT\_0, DM7820\_STDIO\_PORT\_1, DM7820\_STDIO\_PORT\_2, DM7820\_StdIO\_Set\_IO\_Mode(), DM7820\_StdIO\_Set\_Output(), DM7820\_StdIO\_Set\_Periph\_Mode(), DM7820\_STDIO\_STROBE\_1, DM7820\_STDIO\_STROBE\_2, DM7820\_StdIO\_Strobe\_Input(), DM7820\_StdIO\_Strobe\_Mode(), DM7820\_StdIO\_Strobe\_Output(), DM7820\_TMRCTR\_CLOCK\_5\_MHZ, DM7820\_TMRCTR\_CLOCK\_INV\_STROBE\_2, DM7820\_TMRCTR\_CLOCK\_RESERVED, DM7820\_TMRCTR\_COUNT\_MODE\_BCD, DM7820\_TMRCTR\_COUNT\_MODE\_BINARY, DM7820\_TMRCTR\_GATE\_LOGIC\_0, DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_15, DM7820\_TmrCtr\_Get\_Status(), DM7820\_TmrCtr\_Program(), DM7820\_TmrCtr\_Read(), DM7820\_TmrCtr\_Select\_Clock(), DM7820\_TmrCtr\_Select\_Gate(), DM7820\_TMRCTR\_TIMER\_A\_0, DM7820\_TMRCTR\_TIMER\_B\_2, DM7820\_TMRCTR\_WAVEFORM\_EVENT\_CTR, DM7820\_TMRCTR\_WAVEFORM\_HARDWARE\_STROBE, expect\_failure\_and\_check(), expect\_success(), DM7820\_Board\_Descriptor::file\_descriptor, INIT\_BOARD\_ARRAY\_ALLOCATED, INIT\_BOARD\_ARRAY\_OPENED, init\_state, dm7820\_pci\_access\_request::offset, program\_name, dm7820\_ioctl\_argument::readwrite, dm7820\_pci\_access\_request::region, dm7820\_pci\_access\_request::size, and usage().

## 6.15.6 Variable Documentation

### 6.15.6.1 volatile char bad\_device\_name[30] [static]

Path name of DM7820 device file with invalid minor number

Definition at line 134 of file library\_test.c.

Referenced by cleanup().

#### 6.15.6.2 volatile DM7820\_Board\_Descriptor\*\* boards [static]

Array of DM7820 library device descriptors

Definition at line 128 of file library\_test.c.

#### 6.15.6.3 volatile unsigned long device\_count = 1 [static]

Number of devices found when driver was loaded

Definition at line 140 of file library\_test.c.

Referenced by main().

#### 6.15.6.4 volatile initialization\_state\_t init\_state = INIT\_NO\_INITIALIZATION [static]

Program initialization state

Definition at line 122 of file library\_test.c.

Referenced by cleanup(), and main().

#### 6.15.6.5 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 116 of file library\_test.c.

Referenced by main(), and usage().

## 6.16 examples/loopback.c File Reference

Example program which demonstrates how to use the FIFO block interrupts. Note: This example requires a loop-back cable!

```
#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include "dm7820_library.h"
```

### Macros

- `#define BUF_SIZE 0x40000`
- `#define BUF_NUM 16`
- `#define SAMPLES ( ( BUF_SIZE * BUF_NUM ) / 2 )`
- `#define DMA_BUF_SIZE ( BUF_SIZE * BUF_NUM )`

## Functions

- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- void `ISR (dm7820_interrupt_info interrupt_info)`  
*Userspace ISR.*
- void `clean_up` ()  
*Disable and clean up.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* `program_name`
- volatile int `dma_done_interrupts` = 0
- volatile int `fifo0_empty_interrupts` = 0
- `DM7820_Board_Descriptor` \* `board`

### 6.16.1 Detailed Description

Example program which demonstrates how to use the FIFO block interrupts. Note: This example requires a loop-back cable!

```
The Program does the following
1) Allocate a 4MB Block of output data.
2) Allocate a 4MB input buffer.
3) FIFO 0:
  a. Input Clock = PCI Write
  b. Output Clock = Prog Clock 0
  c. In Data = PCI Data
  d. DREQ = Write Request
4) FIFO 1:
  a. Input Clock = Prog Clock 0
  b. Out Clock = PCI Read
  c. In Data = Port 1
  d. DREQ = Read Request
5) Port 0:
  a. All Outputs
  b. Select FIFO 0 as peripheral
6) Prog Clock 0:
  a. Continuous
  b. Clock Source = 25MHz
  c. Period = 3 (results in 6.25 MHz Clock)
7) Setup and starts both DMA's.
8) Wait until FIFO 0 is not empty.
9) Clock first data value to loopback
  a. Change FIFO 0 output clock to PCI Read
  b. Read from FIFO 0 data register
  c. Change FIFO 0 output clock to Prog Clock 0
10) Start Prog Clock 0
11) Wait until DMA1 is complete
13) Test the Data
12) Turn off Prog Clock 0 and the FIFO's
```

Warning: This example program is only intended to run on a system with enough RAM to support the large buffer's which will be allocated. Preferably 128MB minimum.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
```

Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.

-----

Id:

[loopback.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [loopback.c](#).

## 6.16.2 Macro Definition Documentation

### 6.16.2.1 #define BUF\_NUM 16

The number of buffers we want for each DMA channel.

Definition at line 119 of file [loopback.c](#).

Referenced by [main\(\)](#).

### 6.16.2.2 #define BUF\_SIZE 0x40000

The size of the individual buffers we are asking to driver to create for each DMA channel.

Definition at line 113 of file [loopback.c](#).

Referenced by [main\(\)](#).

### 6.16.2.3 #define DMA\_BUF\_SIZE ( BUF\_SIZE \* BUF\_NUM )

The size of the buffer required to handle SAMPLES number of samples.

Definition at line 131 of file [loopback.c](#).

Referenced by [main\(\)](#).

### 6.16.2.4 #define SAMPLES ( ( BUF\_SIZE \* BUF\_NUM ) / 2 )

The number of samples that can be held by the list of buffers we created.

Definition at line 125 of file [loopback.c](#).

Referenced by [main\(\)](#).

## 6.16.3 Function Documentation

### 6.16.3.1 void ISR ( dm7820\_interrupt\_info *interrupt\_info* )

Userspace ISR.

#### Parameters

<i>interrupt_info</i>	Information about the interrupt, returned by the kernel driver
-----------------------	--

Definition at line 174 of file [loopback.c](#).

References [DM7820\\_INTERRUPT\\_FIFO\\_0\\_EMPTY](#), [DM7820\\_INTERRUPT\\_FIFO\\_1\\_DMA\\_DONE](#), [DM7820\\_Return\\_Status](#), [dma\\_done\\_interrupts](#), [\\_dm7820\\_interrupt\\_info::error](#), [fifo0\\_empty\\_interrupts](#), and [\\_dm7820\\_interrupt\\_info::source](#).

### 6.16.3.2 `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 276 of file loopback.c.

References BUF\_NUM, BUF\_SIZE, clean\_up(), DM7820\_DMA\_DEMAND\_ON\_DM7820\_TO\_PCI, DM7820\_DMA\_DEMAND\_ON\_PCI\_TO\_DM7820, DM7820\_FIFO\_0\_DATA\_INPUT\_PCI\_DATA, DM7820\_FIFO\_1\_DATA\_INPUT\_PORT\_1, DM7820\_FIFO\_DMA\_Configure(), DM7820\_FIFO\_DMA\_Create\_Buffer(), DM7820\_FIFO\_DMA\_Enable(), DM7820\_FIFO\_DMA\_Free\_Buffer(), DM7820\_FIFO\_DMA\_Initialize(), DM7820\_FIFO\_DMA\_Read(), DM7820\_FIFO\_DMA\_REQUEST\_READ, DM7820\_FIFO\_DMA\_REQUEST\_WRITE, DM7820\_FIFO\_DMA\_Write(), DM7820\_FIFO\_Enable(), DM7820\_FIFO\_Get\_Status(), DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_READ, DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_WRITE, DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0, DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0, DM7820\_FIFO\_QUEUE\_0, DM7820\_FIFO\_QUEUE\_1, DM7820\_FIFO\_Read(), DM7820\_FIFO\_Set\_Data\_Input(), DM7820\_FIFO\_Set\_DMA\_Request(), DM7820\_FIFO\_Set\_Input\_Clock(), DM7820\_FIFO\_Set\_Output\_Clock(), DM7820\_FIFO\_STATUS\_EMPTY, DM7820\_General\_Enable\_Interrupt(), DM7820\_General\_Open\_Board(), DM7820\_General\_RemoveISR(), DM7820\_General\_Reset(), DM7820\_General\_SetISRPriority(), DM7820\_INTERRUPT\_FIFO\_0\_EMPTY, DM7820\_INTERRUPT\_FIFO\_1\_EMPTY, DM7820\_PRGCLK\_CLOCK\_0, DM7820\_PRGCLK\_MASTER\_25\_MHZ, DM7820\_PRGCLK\_MODE\_CONTINUOUS, DM7820\_PRGCLK\_MODE\_DISABLED, DM7820\_PrgClk\_Set\_Master(), DM7820\_PrgClk\_Set\_Mode(), DM7820\_PrgClk\_Set\_Period(), DM7820\_PrgClk\_Set\_Start\_Trigger(), DM7820\_PrgClk\_Set\_Stop\_Trigger(), DM7820\_PRGCLK\_START\_IMMEDIATE, DM7820\_PRGCLK\_STOP\_NONE, DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_INPUT, DM7820\_STDIO\_MODE\_PER\_OUT, DM7820\_STDIO\_PERIPH\_FIFO\_0, DM7820\_STDIO\_PERIPH\_FIFO\_1, DM7820\_STDIO\_PORT\_0, DM7820\_STDIO\_PORT\_1, DM7820\_StdIO\_Set\_IO\_Mode(), DM7820\_StdIO\_SetPeriph\_Mode(), DMA\_BUF\_SIZE, dma\_done\_interrupts, fifo0\_empty\_interrupts, ISR(), program\_name, SAMPLES, and usage().

## 6.16.4 Variable Documentation

### 6.16.4.1 `DM7820_Board_Descriptor* board`

Device descriptor

Definition at line 106 of file loopback.c.

### 6.16.4.2 `volatile int dma_done_interrupts = 0`

Variable to count the number of DMA Done interrupts occurring

Definition at line 94 of file loopback.c.

### 6.16.4.3 `volatile int fifo0_empty_interrupts = 0`

Variable to count the number of FIFO0 empty interrupts occurring

Definition at line 100 of file loopback.c.

Referenced by ISR(), and main().

#### 6.16.4.4 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 88 of file loopback.c.

Referenced by main(), and usage().

## 6.17 examples/prgclk\_clocks.c File Reference

Example program which demonstrates how to use the programmable clock block clocks.

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

### Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* [program\\_name](#)

### 6.17.1 Detailed Description

Example program which demonstrates how to use the programmable clock block clocks.

This program sets up programmable clocks 0 and 1 so that they run at the same frequency but approximately 180 degrees out of phase. The phase shift is only approximate because the delays inherent in starting clock 2 via clock 0 and starting clock 1 via clock 2 must be considered when setting clock 2's period. Clock 2 is used to delay clock 1 relative to clock 0.

The programmable clocks are set up as follows:

Clock	Start Trigger	Stop Trigger	Mode	Frequency
0	immediate	none	continuous	50 KHz
1	clock 2	none	continuous	50 KHz
2	clock 0	none	one-shot	100 KHz

The above clocks are set to use the 25 MHz clock as their master clocks.

The standard I/O block port 2 pins are set up to output the programmable clocks so that their frequencies may be measured.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

\$Id: prgclk\_clocks.c 60252 2012-06-04 19:39:05Z rgroner \$

Definition in file [prgclk\\_clocks.c](#).

## 6.17.2 Function Documentation

### 6.17.2.1 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 121 of file [prgclk\\_clocks.c](#).

References [board](#), [DM7820\\_General\\_Close\\_Board\(\)](#), [DM7820\\_General\\_Open\\_Board\(\)](#), [DM7820\\_PRGCLK\\_CLOCK\\_0](#), [DM7820\\_PRGCLK\\_CLOCK\\_1](#), [DM7820\\_PRGCLK\\_CLOCK\\_2](#), [DM7820\\_PRGCLK\\_CLOCK\\_3](#), [DM7820\\_PRGCLK\\_MASTER\\_25\\_MHZ](#), [DM7820\\_PRGCLK\\_MODE\\_CONTINUOUS](#), [DM7820\\_PRGCLK\\_MODE\\_DISABLED](#), [DM7820\\_PRGCLK\\_MODE\\_ONE\\_SHOT](#), [DM7820\\_PrgClk\\_Set\\_Master\(\)](#), [DM7820\\_PrgClk\\_Set\\_Mode\(\)](#), [DM7820\\_PrgClk\\_Set\\_Period\(\)](#), [DM7820\\_PrgClk\\_Set\\_Start\\_Trigger\(\)](#), [DM7820\\_PrgClk\\_Set\\_Stop\\_Trigger\(\)](#), [DM7820\\_PRGCLK\\_START\\_IMMEDIATE](#), [DM7820\\_PRGCLK\\_START\\_PROG\\_CLOCK\\_0](#), [DM7820\\_PRGCLK\\_START\\_PROG\\_CLOCK\\_2](#), [DM7820\\_PRGCLK\\_STOP\\_NONE](#), [DM7820\\_Return\\_Status](#), [DM7820\\_STDIO\\_MODE\\_PER\\_OUT](#), [DM7820\\_STDIO\\_PERIPH\\_CLK\\_OTHER](#), [DM7820\\_STDIO\\_PORT\\_2](#), [DM7820\\_StdIO\\_Set\\_IO\\_Mode\(\)](#), [DM7820\\_StdIO\\_Set\\_Periph\\_Mode\(\)](#), [program\\_name](#), and [usage\(\)](#).

## 6.17.3 Variable Documentation

### 6.17.3.1 char\* *program\_name* [static]

Name of the program as invoked on the command line

Definition at line 65 of file [prgclk\\_clocks.c](#).

Referenced by [main\(\)](#), and [usage\(\)](#).

## 6.18 examples/pwm\_interrupt.c File Reference

Example program which demonstrates how to use the pulse width modulator block interrupts.



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

## Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)

### 6.18.1 Detailed Description

Example program which demonstrates how to use the pulse width modulator block interrupts.

Programmable clock 0 is set up as follows:

Master Clock	Start Trigger	Stop Trigger	Mode	Frequency
=====	=====	=====	=====	=====
25 MHz	immediate	none	continuous	500 Hz

This program configures pulse width modulator 0 as follows: 1) master clock set to programmable clock 0, 2) period set to obtain frequency of 1 Hz, 3) width master clock set to programmable clock 0, and 4) output A width initially set to obtain 20% positive duty cycle.

The standard I/O block port 2 pins are set up to output the pulse width modulators so that PWM 0 frequency and duty cycle may be measured.

With the setup indicated about, PWM 0 generates an interrupt once a second.

This program uses pulse width modulator 0 interrupts and waits for the interrupts to occur. 20 such interrupts are waited on and then the program exits. After 10 interrupts occur, the PWM 0 output A width is changed to obtain 80% positive duty cycle.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[pwm\\_interrupt.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [pwm\\_interrupt.c](#).

## 6.18.2 Function Documentation

### 6.18.2.1 `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 131 of file `pwm_interrupt.c`.

References `board`, `DM7820_General_Close_Board()`, `DM7820_General_Enable_Interrupt()`, `DM7820_General_Get_Interrupt_Status()`, `DM7820_General_Open_Board()`, `DM7820_General_Reset()`, `DM7820_INTERRUPT_PWM_0`, `DM7820_PRGCLK_CLOCK_0`, `DM7820_PRGCLK_MASTER_25_MHZ`, `DM7820_PRGCLK_MODE_CONTINUOUS`, `DM7820_PRGCLK_MODE_DISABLED`, `DM7820_PrgClk_Set_Master()`, `DM7820_PrgClk_Set_Mode()`, `DM7820_PrgClk_Set_Period()`, `DM7820_PrgClk_Set_Start_Trigger()`, `DM7820_PrgClk_Set_Stop_Trigger()`, `DM7820_PRGCLK_START_IMMEDIATE`, `DM7820_PRGCLK_STOP_NONE`, `DM7820_PWM_Enable()`, `DM7820_PWM_OUTPUT_A`, `DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0`, `DM7820_PWM_Set_Period()`, `DM7820_PWM_Set_Period_Master()`, `DM7820_PWM_Set_Width()`, `DM7820_PWM_Set_Width_Master()`, `DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0`, `DM7820_Return_Status`, `DM7820_STDIO_MODE_PER_OUT`, `DM7820_STDIO_PERIPH_PWM`, `DM7820_STDIO_PORT_2`, `DM7820_StdIO_Set_IO_Mode()`, `DM7820_StdIO_SetPeriph_Mode()`, `program_name`, `_dm7820_interrupt_info::source`, and `usage()`.

## 6.18.3 Variable Documentation

### 6.18.3.1 `char* program_name [static]`

Name of the program as invoked on the command line

Definition at line 68 of file `pwm_interrupt.c`.

Referenced by `main()`, and `usage()`.

## 6.19 examples/pwm\_measure.c File Reference

This program demonstrates how an incoming pulse width can be measured using the DM7820.

```

#include <errno.h>
#include <error.h>
#include <getopt.h>
#include <limits.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include "dm7820_library.h"

```

## Macros

- `#define BUF_SIZE 0x4000`
- `#define BUF_NUM 8`
- `#define SAMPLES ((BUF_SIZE * BUF_NUM)/2)`
- `#define DMA_BUF_SIZE (BUF_SIZE * BUF_NUM)`
- `#define UTC_RATE 2`

## Functions

- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- void `ISR` (dm7820\_interrupt\_info interrupt\_info)  
*Userspace ISR.*
- void `clean_up` ()  
*Disable and clean up.*
- static void `sigint_handler` (int signal\_number)  
*Signal handler for SIGINT Control-C keyboard interrupt.*
- void `do_digital_io` ()  
*Perform all digital I/O port initialization.*
- void `do_incenc` ()  
*Perform all incremental encoder initialization.*
- void `do_8254` ()  
*Perform all 8254 Timer/Counter initialization.*
- void `do_pwm` ()  
*Perform all PWM initialization.*
- void `do_fifo` ()  
*Perform all FIFO initialization.*
- void `do_dma` ()  
*Perform all DMA initialization.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

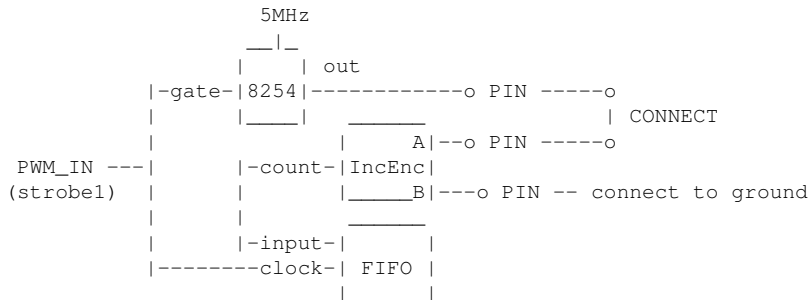
## Variables

- static char \* `program_name`
- volatile int `dma_done_interrupts` = 0
- `DM7820_Board_Descriptor` \* `board`
- uint8\_t `exit_program` = 0

## 6.19.1 Detailed Description

This program demonstrates how an incoming pulse width can be measured using the DM7820.

The example program implements the block diagram drawn below



The pulse width coming from the position sensor should be connected to strobe1 (Pin 2 on CN11). This example uses PWM0 Output A (Pin 15 CN10) to emulate the position sensor input on strobe1.

The strobe1 input is then internally used as the gate input for Timer/Counter A0. Timer/Counter A0 out is then externally used as the input for Incremental Encoder 1 Channel A (connect Pin 11 CN10 to Pin 47 CN11). The incremental encoder's count value is then inserted into the FIFO.

Encoder count values are then read from the FIFO. Taking a difference from those values will give the number of 8254 User Timer/Counter periods that have occurred. If that period is known value, the length of the Duty Cycle can then be calculated.

Warning: This example program is only intended to run on a system with enough RAM to support the large buffer's which will be allocated. Preferably 128MB minimum.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[dma\\_capture\\_buffers.c](#) 33597 2008-11-26 16:46:30Z wtate

Definition in file [pwm\\_measure.c](#).

## 6.19.2 Macro Definition Documentation

### 6.19.2.1 #define BUF\_NUM 8

The number of buffers we want for each DMA channel.

Definition at line 118 of file [pwm\\_measure.c](#).

Referenced by [do\\_dma\(\)](#).

### 6.19.2.2 #define BUF\_SIZE 0x4000

The size of the individual buffers we are asking to driver to create for each DMA channel.

Definition at line 112 of file pwm\_measure.c.

Referenced by do\_dma(), and main().

#### 6.19.2.3 #define DMA\_BUF\_SIZE (BUF\_SIZE \* BUF\_NUM)

The total space available in the device for DMA at any one time (4MB).

Definition at line 130 of file pwm\_measure.c.

Referenced by main().

#### 6.19.2.4 #define SAMPLES ((BUF\_SIZE \* BUF\_NUM)/2)

The number of samples that can be held by the list of buffers we created.

Definition at line 124 of file pwm\_measure.c.

#### 6.19.2.5 #define UTC\_RATE 2

User Timer/Counter divisor, the smaller this number is the better your resolution for measuring PW is.

Definition at line 135 of file pwm\_measure.c.

Referenced by do\_8254(), and main().

### 6.19.3 Function Documentation

#### 6.19.3.1 void ISR ( dm7820\_interrupt\_info *interrupt\_info* )

Userspace ISR.

##### Parameters

<i>interrupt_info</i>	Information about the interrupt, returned by the kernel driver
-----------------------	--

Definition at line 178 of file pwm\_measure.c.

References DM7820\_INTERRUPT\_FIFO\_1\_DMA\_DONE, DM7820\_Return\_Status, dma\_done\_interrupts, \_dm7820\_interrupt\_info::error, and \_dm7820\_interrupt\_info::source.

#### 6.19.3.2 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

##### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

##### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 582 of file pwm\_measure.c.

References BUF\_SIZE, clean\_up(), DM7820\_FIFO\_DMA\_Create\_Buffer(), DM7820\_FIFO\_DMA\_Enable(), DM7820\_FIFO\_DMA\_Free\_Buffer(), DM7820\_FIFO\_DMA\_Read(), DM7820\_FIFO\_Enable(), DM7820\_FIFO\_QUEUE\_1, DM7820\_General\_Enable\_Interrupt(), DM7820\_General\_Open\_Board(), DM7820\_General\_Remove\_ISR(), DM7820\_General\_Reset(), DM7820\_General\_SetISRPriority(), DM7820\_INTERRUPT\_FIFO\_1\_EMPTY, DM7820\_Return\_Status, DMA\_BUF\_SIZE, dma\_done\_interrupts, do\_8254(), do\_digital\_io(), do\_dma(), do\_fifo(), do\_incenc(), do\_pwm(), exit\_program, ISR(), program\_name, sigint\_handler(), usage(), and UTC\_RATE.

#### 6.19.3.3 static void sigint\_handler ( int *signal\_number* ) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

##### Parameters

<i>signal_number</i>	Signal number passed in from the kernel.
----------------------	--

##### Warning

One must be extremely careful about what functions are called from a signal handler.

Definition at line 246 of file pwm\_measure.c.

References exit\_program.

Referenced by main().

## 6.19.4 Variable Documentation

#### 6.19.4.1 DM7820\_Board\_Descriptor\* board

Device descriptor

Definition at line 99 of file pwm\_measure.c.

#### 6.19.4.2 volatile int dma\_done\_interrupts = 0

Variable to count the number of DMA Done interrupts occurring

Definition at line 93 of file pwm\_measure.c.

#### 6.19.4.3 uint8\_t exit\_program = 0

Flag indicating user intent to exit the program

Definition at line 105 of file pwm\_measure.c.

Referenced by main(), and sigint\_handler().

#### 6.19.4.4 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 87 of file pwm\_measure.c.

Referenced by main(), and usage().

## 6.20 examples/pwm\_modulators.c File Reference

Example program which demonstrates how to use the pulse width modulator block modulators.

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

## Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)

### 6.20.1 Detailed Description

Example program which demonstrates how to use the pulse width modulator block modulators.

This program sets up the pulse width modulators as follows:

PWM	Output	Duty Cycle
=====		
0	A	20%
	B	40%
	C	60%
	D	80%
1	A	20%
	B	40%
	C	60%
	D	80%

PWMs 0 and 1 are set to use the 25 MHz clock as their period and width master clocks. The period of PWM 0 is set to provide a frequency of 1 MHz. The period of PWM 1 is set to provide a frequency of 100 KHz.

The standard I/O block port 2 pins are set up to output the pulse width modulators so that their frequencies and duty cycles may be measured.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.
```

```
This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

```
$Id: pwm_modulators.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file [pwm\\_modulators.c](#).

## 6.20.2 Function Documentation

### 6.20.2.1 `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 120 of file `pwm_modulators.c`.

References `board`, `DM7820_General_Close_Board()`, `DM7820_General_Open_Board()`, `DM7820_PWM_Enable()`, `DM7820_PWM_OUTPUT_A`, `DM7820_PWM_OUTPUT_B`, `DM7820_PWM_OUTPUT_C`, `DM7820_PWM_OUTPUT_D`, `DM7820_PWM_PERIOD_MASTER_25_MHZ`, `DM7820_PWM_Set_Period()`, `DM7820_PWM_Set_Period_Master()`, `DM7820_PWM_Set_Width()`, `DM7820_PWM_Set_Width_Master()`, `DM7820_PWM_WIDTH_MASTER_25_MHZ`, `DM7820_Return_Status`, `DM7820_STDIO_MODE_PER_OUT`, `DM7820_STDIO_PERIPH_PWM`, `DM7820_STDIO_PORT_2`, `DM7820_StdIO_Set_IO_Mode()`, `DM7820_StdIO_Set_Periph_Mode()`, `program_name`, and `usage()`.

## 6.20.3 Variable Documentation

### 6.20.3.1 `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 64 of file `pwm_modulators.c`.

Referenced by `main()`, and `usage()`.

## 6.21 `examples/stdio_digital_io.c` File Reference

Example program which demonstrates how to use the standard I/O block digital I/O.

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

## Functions

- static void `usage` (void)

*Print information on stderr about how the program is to be used. After doing so, the program is exited.*



- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)

### 6.21.1 Detailed Description

Example program which demonstrates how to use the standard I/O block digital I/O.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

Id:

[stdio\\_digital\\_io.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [stdio\\_digital\\_io.c](#).

### 6.21.2 Function Documentation

#### 6.21.2.1 int main ( int argument\_count, char \*\* arguments )

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 107 of file [stdio\\_digital\\_io.c](#).

References [board](#), [DM7820\\_General\\_Close\\_Board\(\)](#), [DM7820\\_General\\_Open\\_Board\(\)](#), [DM7820\\_Return\\_Status](#), [DM7820\\_StdIO\\_Get\\_Input\(\)](#), [DM7820\\_STDIO\\_MODE\\_INPUT](#), [DM7820\\_STDIO\\_MODE\\_OUTPUT](#), [DM7820\\_StdIO\\_Set\\_IO\\_Mode\(\)](#), [DM7820\\_StdIO\\_Set\\_Output\(\)](#), [program\\_name](#), and [usage\(\)](#).

### 6.21.3 Variable Documentation

#### 6.21.3.1 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 43 of file [stdio\\_digital\\_io.c](#).

Referenced by [main\(\)](#), and [usage\(\)](#).

## 6.22 examples/stdio\_peripheral\_output.c File Reference

Example program which demonstrates how to select peripheral outputs for standard I/O block digital I/O ports.

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

### Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* [program\\_name](#)

### 6.22.1 Detailed Description

Example program which demonstrates how to select peripheral outputs for standard I/O block digital I/O ports.

#### Note

This program does not set up the selected peripheral output. It merely programs the digital I/O ports so that a peripheral can be output.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

```
$Id: stdio_peripheral_output.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file [stdio\\_peripheral\\_output.c](#).

### 6.22.2 Function Documentation

#### 6.22.2.1 int main ( int argument\_count, char \*\* arguments )

Main program code.

## Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

## Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 125 of file stdio\_peripheral\_output.c.

References board, DM7820\_General\_Close\_Board(), DM7820\_General\_Open\_Board(), DM7820\_Return\_Status, DM7820\_STDIO\_MODE\_PER\_OUT, DM7820\_STDIO\_PERIPH\_CLK\_OTHER, DM7820\_STDIO\_PERIPH\_FIFO\_0, DM7820\_STDIO\_PERIPH\_FIFO\_1, DM7820\_STDIO\_PERIPH\_PWM, DM7820\_StdIO\_Set\_IO\_Mode(), DM7820\_StdIO\_Set\_Periph\_Mode(), program\_name, and usage().

### 6.22.3 Variable Documentation

#### 6.22.3.1 char\* program\_name [static]

Name of the program as invoked on the command line

Definition at line 48 of file stdio\_peripheral\_output.c.

Referenced by main(), and usage().

## 6.23 examples/stdio\_strobe\_signal.c File Reference

Example program which demonstrates how to use the standard I/O block strobe signals.

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

### Functions

- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

### Variables

- static char \* [program\\_name](#)

### 6.23.1 Detailed Description

Example program which demonstrates how to use the standard I/O block strobe signals.

#### Note

This program uses standard I/O block strobe signals 1 and 2. Each signal is used to change the state of the other. Therefore, CN11 pin 2 should be connected to CN10 pin 2.

```
-----
This file and its contents are copyright (C) RTD Embedded Technologies,
Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License
Agreement. For a copy of this agreement, refer to the file LICENSE.TXT
(which should be included with this software) or contact RTD Embedded
Technologies, Inc.
-----
```

```
$Id: stdio_strobe_signal.c 60252 2012-06-04 19:39:05Z rgroner $
```

Definition in file [stdio\\_strobe\\_signal.c](#).

### 6.23.2 Function Documentation

#### 6.23.2.1 `int main ( int argument_count, char ** arguments )`

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 104 of file `stdio_strobe_signal.c`.

References `board`, `DM7820_General_Close_Board()`, `DM7820_General_Open_Board()`, `DM7820_Return_Status`, `DM7820_STDIO_STROBE_1`, `DM7820_STDIO_STROBE_2`, `DM7820_StdIO_Strobe_Input()`, `DM7820_StdIO_-Strobe_Mode()`, `DM7820_StdIO_Strobe_Output()`, `program_name`, and `usage()`.

### 6.23.3 Variable Documentation

#### 6.23.3.1 `char* program_name [static]`

Name of the program as invoked on the command line

Definition at line 48 of file `stdio_strobe_signal.c`.

Referenced by `main()`, and `usage()`.

## 6.24 examples/tmrctr\_interrupt.c File Reference

Example program which demonstrates how to use 8254 timer/counter block timer interrupts.

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

## Functions

- static void [sigint\\_handler](#) (int signal\_number)  
*Signal handler for SIGINT Control-C keyboard interrupt.*
- static void [usage](#) (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- void [ISR](#) ([dm7820\\_interrupt\\_info](#) interrupt\_info)  
*Userspace ISR.*
- int [main](#) (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* [program\\_name](#)
- static volatile int [interrupts](#) = 0
- static volatile sig\_atomic\_t [exit\\_program](#) = 0

### 6.24.1 Detailed Description

Example program which demonstrates how to use 8254 timer/counter block timer interrupts.

Timers A0 and A1 are configured as follows:

Timer	Input Clock	Count Mode	Waveform Mode	Frequency
A0	5 MHz	binary	square wave	100 Hz
A1	A0	binary	square wave	.5 Hz

Each timer has its gate set to logic 1 to enable counting.

With the setup indicated above, timers A0 and A1 are cascaded. Timer A1 generates an interrupt once every two seconds.

The program uses timer A1 interrupts and waits for the interrupts to occur. Ten such interrupts are waited on and then the program exits.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.

Id:

[tmrctr\\_interrupt.c](#) 79285 2014-05-21 20:37:42Z rgroner

Definition in file [tmrctr\\_interrupt.c](#).

## 6.24.2 Function Documentation

### 6.24.2.1 void ISR ( *dm7820\_interrupt\_info interrupt\_info* )

Userspace ISR.

#### Parameters

<i>interrupt_info</i>	Information about the interrupt, returned by the kernel driver
-----------------------	--

Definition at line 155 of file [tmrctr\\_interrupt.c](#).

References [DM7820\\_INTERRUPT\\_TMRCTR\\_A\\_1](#), [DM7820\\_Return\\_Status](#), [\\_dm7820\\_interrupt\\_info::error](#), [interrupts](#), and [\\_dm7820\\_interrupt\\_info::source](#).

### 6.24.2.2 int main ( int *argument\_count*, char \*\* *arguments* )

Main program code.

#### Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

#### Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 201 of file [tmrctr\\_interrupt.c](#).

References [board](#), [DM7820\\_General\\_Close\\_Board\(\)](#), [DM7820\\_General\\_Enable\\_Interrupt\(\)](#), [DM7820\\_General\\_Open\\_Board\(\)](#), [DM7820\\_General\\_RemoveISR\(\)](#), [DM7820\\_General\\_Reset\(\)](#), [DM7820\\_General\\_SetISRPriority\(\)](#), [DM7820\\_INTERRUPT\\_TMRCTR\\_A\\_1](#), [DM7820\\_Return\\_Status](#), [DM7820\\_TMRCTR\\_CLOCK\\_5\\_MHZ](#), [DM7820\\_TMRCTR\\_CLOCK\\_8254\\_A\\_0](#), [DM7820\\_TMRCTR\\_COUNT\\_MODE\\_BINARY](#), [DM7820\\_TMRCTR\\_GATE\\_LOGIC\\_0](#), [DM7820\\_TMRCTR\\_GATE\\_LOGIC\\_1](#), [DM7820\\_TmrCtr\\_Program\(\)](#), [DM7820\\_TmrCtr\\_Select\\_Clock\(\)](#), [DM7820\\_TmrCtr\\_Select\\_Gate\(\)](#), [DM7820\\_TMRCTR\\_TIMER\\_A\\_0](#), [DM7820\\_TMRCTR\\_TIMER\\_A\\_1](#), [DM7820\\_TMRCTR\\_WAVEFORM\\_SQUARE\\_WAVE](#), [exit\\_program](#), [interrupts](#), [ISR\(\)](#), [program\\_name](#), [sigint\\_handler\(\)](#), and [usage\(\)](#).

### 6.24.2.3 static void sigint\_handler ( int *signal\_number* ) [static]

Signal handler for SIGINT Control-C keyboard interrupt.

#### Parameters

<i>signal_number</i>	Signal number passed in from kernel.
----------------------	--------------------------------------

#### Warning

One must be extremely careful about what functions are called from a signal handler. `printf()` and related functions are considered unsafe for use in signal handlers. Therefore, this function uses `write()` instead.

Definition at line 98 of file tmrctr\_interrupt.c.

References `exit_program`.

Referenced by `main()`.

### 6.24.3 Variable Documentation

#### 6.24.3.1 `volatile sig_atomic_t exit_program = 0` `[static]`

Flag used by SIGINT signal handler to tell main program to exit because the user hit Control-C

Definition at line 75 of file tmrctr\_interrupt.c.

#### 6.24.3.2 `volatile int interrupts = 0` `[static]`

Variable to count the number of DMA Done interrupts occurring

Definition at line 68 of file tmrctr\_interrupt.c.

#### 6.24.3.3 `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 62 of file tmrctr\_interrupt.c.

Referenced by `main()`, and `usage()`.

## 6.25 examples/tmrctr\_status.c File Reference

Example program which demonstrates how to get 8254 timer/counter block timer status when not using interrupts.

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

### Macros

- `#define` `TIMER_A0_DIVISOR` 50000
- `#define` `TIMER_A1_DIVISOR` 2
- `#define` `TIMER_A1_FREQUENCY` 50
- `#define` `TIMER_A2_DIVISOR` 1000

### Functions

- static void `disable_timers` (`DM7820_Board_Descriptor *board`)  
*Disable all 8254 timer/counters by setting their gates to logic zero.*
- static void `usage` (void)

*Print information on stderr about how the program is to be used. After doing so, the program is exited.*

- `int main (int argument_count, char **arguments)`

*Main program code.*

## Variables

- `static char * program_name`

### 6.25.1 Detailed Description

Example program which demonstrates how to get 8254 timer/counter block timer status when not using interrupts.

Timers A0, A1, and A2 are configured as follows:

Timer	Input Clock	Count Mode	Waveform Mode	Frequency
A0	5 MHz	binary	square wave	100 Hz
A1	A0	binary	square wave	50 Hz
A2	A1	binary	event counter	

Each timer has its gate set to logic 1 to enable counting.

With the setup indicated above, timers A0, A1, and A2 are cascaded and timer A2 is decremented every tick of timer A1, i.e. every 20 milliseconds.

Timer A2 is loaded with a divisor value of 1000. With this value, it will take 20 seconds for timer A2 to count down to zero.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[tmrctr\\_status.c](#) 60252 2012-06-04 19:39:05Z rgroner

Definition in file [tmrctr\\_status.c](#).

### 6.25.2 Macro Definition Documentation

#### 6.25.2.1 #define TIMER\_A0\_DIVISOR 50000

Timer/counter A0 divisor

Definition at line 63 of file [tmrctr\\_status.c](#).

Referenced by [main\(\)](#).

#### 6.25.2.2 #define TIMER\_A1\_DIVISOR 2

Timer/counter A1 divisor

Definition at line 69 of file [tmrctr\\_status.c](#).

Referenced by [main\(\)](#).



**6.25.2.3 #define TIMER\_A1\_FREQUENCY 50**

Timer/counter A1 frequency in Hertz

Definition at line 75 of file tmrctr\_status.c.

Referenced by main().

**6.25.2.4 #define TIMER\_A2\_DIVISOR 1000**

Timer/counter A2 divisor

Definition at line 81 of file tmrctr\_status.c.

Referenced by main().

**6.25.3 Function Documentation****6.25.3.1 static void disable\_timers ( DM7820\_Board\_Descriptor \* board ) [static]**

Disable all 8254 timer/counters by setting their gates to logic zero.

**Parameters**

<i>board</i>	Address of device's library board descriptor.
--------------	---

Definition at line 109 of file tmrctr\_status.c.

References DM7820\_Return\_Status, DM7820\_TMRCTR\_GATE\_LOGIC\_0, DM7820\_TmrCtr\_Select\_Gate(), DM7820\_TMRCTR\_TIMER\_A\_0, DM7820\_TMRCTR\_TIMER\_A\_1, DM7820\_TMRCTR\_TIMER\_A\_2, DM7820\_TMRCTR\_TIMER\_B\_0, DM7820\_TMRCTR\_TIMER\_B\_1, and DM7820\_TMRCTR\_TIMER\_B\_2.

Referenced by main().

**6.25.3.2 int main ( int argument\_count, char \*\* arguments )**

Main program code.

**Parameters**

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

**Return values**

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 209 of file tmrctr\_status.c.

References board, disable\_timers(), DM7820\_General\_Close\_Board(), DM7820\_General\_Open\_Board(), DM7820\_Return\_Status, DM7820\_TMRCTR\_CLOCK\_5\_MHZ, DM7820\_TMRCTR\_CLOCK\_8254\_A\_0, DM7820\_TMRCTR\_CLOCK\_8254\_A\_1, DM7820\_TMRCTR\_COUNT\_MODE\_BINARY, DM7820\_TMRCTR\_GATE\_LOGIC\_1, DM7820\_TmrCtr\_Get\_Status(), DM7820\_TmrCtr\_Program(), DM7820\_TmrCtr\_Read(), DM7820\_TmrCtr\_Select\_Clock(), DM7820\_TmrCtr\_Select\_Gate(), DM7820\_TMRCTR\_TIMER\_A\_0, DM7820\_TMRCTR\_TIMER\_A\_1, DM7820\_TMRCTR\_TIMER\_A\_2, DM7820\_TMRCTR\_WAVEFORM\_EVENT\_CTR, DM7820\_TMRCTR\_WAVEFORM\_SQUARE\_WAVE, program\_name, TIMER\_A0\_DIVISOR, TIMER\_A1\_DIVISOR, TIMER\_A1\_FREQUENCY, TIMER\_A2\_DIVISOR, and usage().

## 6.25.4 Variable Documentation

### 6.25.4.1 `char* program_name` [static]

Name of the program as invoked on the command line

Definition at line 91 of file `tmrctr_status.c`.

Referenced by `main()`, and `usage()`.

## 6.26 examples/tmrctr\_timers.c File Reference

Example program which demonstrates how to use the 8254 timer/counter block timers.

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

## Functions

- static void `sigint_handler` (int signal\_number)  
*Signal handler for SIGINT Control-C keyboard interrupt.*
- static void `usage` (void)  
*Print information on stderr about how the program is to be used. After doing so, the program is exited.*
- int `main` (int argument\_count, char \*\*arguments)  
*Main program code.*

## Variables

- static char \* `program_name`
- static volatile sig\_atomic\_t `exit_program` = 0

### 6.26.1 Detailed Description

Example program which demonstrates how to use the 8254 timer/counter block timers.

All timers are set to count in binary mode. Timers A0, A1, A2, B0, and B1 are set to square wave generator waveform mode. Timer B2 is set to event counter waveform mode. Each timer has its gate set to logic 1 to enable counting. The input clocks and frequencies are set as follows:

Timer	Input Clock	Frequency
=====		
A0	5 MHz	500 KHz
A1	A0	50 KHz
A2	A1	5 KHz
B0	A2	500 Hz
B1	B0	50 Hz

B2                      B1

With the setup indicated above, all six timers are cascaded and timer B2 is decremented every tick of timer B1, i.e. every 20 milliseconds. Thus, timer B2 indicates the length of time that timer B1 has been running.

-----  
This file and its contents are copyright (C) RTD Embedded Technologies, Inc. All Rights Reserved.

This software is licensed as described in the RTD End-User Software License Agreement. For a copy of this agreement, refer to the file LICENSE.TXT (which should be included with this software) or contact RTD Embedded Technologies, Inc.  
-----

Id:

[tmrctr\\_timers.c](#) 79285 2014-05-21 20:37:42Z rgroner

Definition in file [tmrctr\\_timers.c](#).

## 6.26.2 Function Documentation

### 6.26.2.1 `int main ( int argument_count, char ** arguments )`

Main program code.

Parameters

<i>argument_count</i>	Number of command line arguments passed to executable.
<i>arguments</i>	Address of array containing command line arguments.

Return values

<i>EXIT_SUCCESS</i>	Success.
<i>EXIT_FAILURE</i>	Failure.

Definition at line 165 of file `tmrctr_timers.c`.

References `board`, `DM7820_General_Close_Board()`, `DM7820_General_Open_Board()`, `DM7820_Return_Status`, `DM7820_TMRCTR_CLOCK_5_MHZ`, `DM7820_TMRCTR_CLOCK_8254_A_0`, `DM7820_TMRCTR_CLOCK_8254_A_1`, `DM7820_TMRCTR_CLOCK_8254_A_2`, `DM7820_TMRCTR_CLOCK_8254_B_0`, `DM7820_TMRCTR_CLOCK_8254_B_1`, `DM7820_TMRCTR_COUNT_MODE_BINARY`, `DM7820_TMRCTR_GATE_LOGIC_0`, `DM7820_TMRCTR_GATE_LOGIC_1`, `DM7820_TmrCtr_Program()`, `DM7820_TmrCtr_Read()`, `DM7820_TmrCtr_Select_Clock()`, `DM7820_TmrCtr_Select_Gate()`, `DM7820_TMRCTR_TIMER_A_0`, `DM7820_TMRCTR_TIMER_A_1`, `DM7820_TMRCTR_TIMER_A_2`, `DM7820_TMRCTR_TIMER_B_0`, `DM7820_TMRCTR_TIMER_B_1`, `DM7820_TMRCTR_TIMER_B_2`, `DM7820_TMRCTR_WAVEFORM_EVENT_CTR`, `DM7820_TMRCTR_WAVEFORM_SQUARE_WAVE`, `exit_program`, `program_name`, `sigint_handler()`, and `usage()`.

### 6.26.2.2 `static void sigint_handler ( int signal_number )` `[static]`

Signal handler for SIGINT Control-C keyboard interrupt.

Parameters

<i>signal_number</i>	Signal number passed in from kernel.
----------------------	--------------------------------------

**Warning**

One must be extremely careful about what functions are called from a signal handler. `printf()` and related functions are considered unsafe for use in signal handlers. Therefore, this function uses `write()` instead.

Definition at line 95 of file `tmrctr_timers.c`.

References `exit_program`.

Referenced by `main()`.

**6.26.3 Variable Documentation****6.26.3.1** `volatile sig_atomic_t exit_program = 0` `[static]`

Flag used by SIGINT signal handler to tell main program to exit because the user hit Control-C

Definition at line 72 of file `tmrctr_timers.c`.

**6.26.3.2** `char* program_name` `[static]`

Name of the program as invoked on the command line

Definition at line 65 of file `tmrctr_timers.c`.

Referenced by `main()`, and `usage()`.

**6.27 include/dm7820\_driver.h File Reference**

Definitions for the DM7820 driver.

```
#include <linux/fs.h>
#include <linux/list.h>
#include <linux/pci.h>
#include <linux/spinlock.h>
#include <linux/types.h>
#include "dm7820_ioctl.h"
#include "dm7820_types.h"
```

**Data Structures**

- struct [dm7820\\_pci\\_region](#)  
*DM7820 PCI region descriptor. This structure holds information about one of a device's PCI memory regions.*
- struct [dm7820\\_dma\\_descriptor\\_t](#)  
*DM7820 DMA buffer descriptor. This structure holds allocation information for a single DMA buffer.*
- struct [dm7820\\_dma\\_list\\_item\\_t](#)  
*DM7820 DMA buffer list item.*
- struct [dm7820\\_device\\_descriptor](#)  
*DM7820 device descriptor. This structure holds information about a device needed by the kernel.*
- struct [dm7820\\_interrupt\\_status\\_source](#)  
*Interrupt source information for a single Interrupt Status Register bit.*

## Macros

- `#define DM7820_DEVICE_NAME_LENGTH 22`  
*Maximum number of characters in device's name.*
- `#define DM7820_PCI_DEVICE_ID 0x7820`  
*DM7820 PCI device ID.*
- `#define RTD_PCI_VENDOR_ID 0x1435`  
*RTD Embedded Technologies PCI vendor ID.*
- `#define DM7820_PCI_REGIONS PCI_ROM_RESOURCE`  
*Number of standard PCI regions.*
- `#define DM7820_FIFO_CHANNELS 2`  
*Number of FIFO channels per device.*
- `#define DM7820_MAX_DMA_BUFFER_SIZE 0x40000`  
*Maximum size in bytes of any DMA buffer.*
- `#define DM7820_MAX_DMA_BUFFER_COUNT 16`  
*Maximum number of DMA buffers per DMA/FIFO channel.*
- `#define DM7820_INT_QUEUE_SIZE 0x10`  
*Maximum number of entries in the interrupt status queue;.*

## Typedefs

- `typedef enum dm7820_pci_region_access_dir dm7820_pci_region_access_dir_t`  
*Standard PCI region access direction type.*
- `typedef struct dm7820_pci_region dm7820_pci_region_t`  
*DM7820 PCI region descriptor type.*
- `typedef struct dm7820_device_descriptor dm7820_device_descriptor_t`  
*DM7820 device descriptor type.*
- `typedef struct dm7820_interrupt_status_source dm7820_interrupt_status_source_t`  
*Interrupt Status Register bit interrupt source information type.*

## Enumerations

- `enum dm7820_pci_region_access_dir { DM7820_PCI_REGION_ACCESS_READ = 0, DM7820_PCI_REGION_ACCESS_WRITE }`  
*Direction of access to standard PCI region.*

## Functions

- `static void dm7820_access_pci_region (const dm7820_device_descriptor_t *dm7820_device, dm7820_pci_region_access_request_t *pci_request, dm7820_pci_region_access_dir_t direction)`  
*Read from or write to one of the standard PCI regions.*
- `static int dm7820_allocate_irq (dm7820_device_descriptor_t *dm7820_device, const struct pci_dev *pci_device)`  
*Allocate an interrupt line for a DM7820 device.*
- `static void dm7820_disable_all_interrupts (const dm7820_device_descriptor_t *dm7820_device)`  
*Disable all non-PLX interrupts for the specified DM7820 device.*
- `static void dm7820_enable_plx_interrupts (const dm7820_device_descriptor_t *dm7820_device, uint8_t enable)`

- Disable or enable PLX interrupts for the specified DM7820 device.*

  - static void `dm7820_free_dma_mappings` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)

*Free all coherent/consistent DMA mappings for the given DMA/FIFO channel on the specified DM7820 device.*
- static int `dm7820_get_interrupt_status` (`dm7820_device_descriptor_t` \*dm7820\_device, unsigned long ioctl\_param)

*Get interrupt status for the specified DM7820 device, optionally waiting for an interrupt to occur before returning the status.*
- static void `dm7820_get_pci_master_status` (`dm7820_device_descriptor_t` \*dm7820\_device, uint8\_t \*pci\_master)

*Determine whether or not a device is PCI master capable.*
- static void `dm7820_initialize_device_descriptor` (`dm7820_device_descriptor_t` \*dm7820\_device)

*Initialize the device descriptor for the specified DM7820 device.*
- static int `dm7820_initialize_dma` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_ioctl_argument_t` \*ioctl\_argument)

*Initialize DMA for the specified DM7820 device.*
- dma\_addr\_t `dm7820_get_buffer_phy_addr` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)

*Returns the physical address of the next available DMA buffer.*
- static int `dm7820_dma_read` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_ioctl_argument_t` \*ioctl\_argument)

*Read from DMA buffer to copy to user.*
- static int `dm7820_dma_write` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_ioctl_argument_t` \*ioctl\_argument)

*Write to DMA buffer.*
- static int `dm7820_dma_stop` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)

*Stops a DMA transfer on the specified channel if one is currently running.*
- static int `dm7820_dma_pause` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)

*Pause DMA – Used by the STOP\_DMA function.*
- static int `dm7820_dma_check_xfer` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_fifo_queue` fifo)

*Checks if there is a transfer currently underway for the specified DMA/FIFO channel.*
- static void `dm7820_initialize_hardware` (const `dm7820_device_descriptor_t` \*dm7820\_device)

*Initialize the specified DM7820 device.*
- static void `dm7820_int_queue_add` (`dm7820_device_descriptor_t` \*dm7820\_device, `dm7820_interrupt_source` source)

*Add an interrupt source to the queue.*
- static `dm7820_interrupt_info` `dm7820_dequeue_interrupt` (`dm7820_device_descriptor_t` \*dm7820\_device)

*Remove an interrupt from the front of the queue.*
- static irqreturn\_t `dm7820_interrupt_handler` (int irq\_number, void \*device\_id)

*DM7820 device interrupt handler.*
- static long `dm7820_ioctl` (struct file \*file, unsigned int request\_code, unsigned long ioctl\_param)

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

*Perform all actions necessary to initialize the DM7820 driver and devices.*
- static int `dm7820_modify_pci_region` (`dm7820_device_descriptor_t` \*dm7820\_device, unsigned long ioctl\_param)

*Read an unsigned value from one of a device's PCI regions, modify certain bits in the value, and then write it back to the region.*
- static int `dm7820_open` (struct inode \*inode, struct file \*file)

*Prepare a DM7820 device file to be opened and used.*
- static unsigned int `dm7820_poll` (struct file \*file, struct poll\_table\_struct \*poll\_table)

- Determine whether or not a DM7820 device is readable. This function supports the poll(2) and select(2) system calls.*
- static int [dm7820\\_probe\\_device\\_blocks](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device)
- Probe and set up all functional blocks on a device.*
- static int [dm7820\\_probe\\_devices](#) (uint32\_t \*device\_count, [dm7820\\_device\\_descriptor\\_t](#) \*\*device\_descriptors)
- Probe and set up all DM7820 devices.*
- static int [dm7820\\_process\\_pci\\_regions](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, const struct pci\_dev \*pci\_device)
- For each of the standard PCI regions, get the region's base address and length from kernel PCI resource information set up at boot. Also, remap any memory-mapped region into the kernel's virtual address space.*
- static int [dm7820\\_read\\_pci\\_region](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)
- Read an unsigned value from one of a device's PCI regions.*
- static int [dm7820\\_register\\_char\\_device](#) (int \*major)
- Register the DM7820 character device and request dynamic allocation of a character device major number.*
- static int [dm7820\\_release](#) (struct inode \*inode, struct file \*file)
- Do all processing necessary after the last reference to a DM7820 device file is released elsewhere in the kernel.*
- static void [dm7820\\_release\\_resources](#) (void)
- Release any resources allocated by the driver.*
- static int [dm7820\\_service\\_dma\\_function](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)
- Process user space DMA function request.*
- void [dm7820\\_unload](#) (void)
- Perform all actions necessary to deinitialize the DM7820 driver and devices.*
- static int [dm7820\\_unregister\\_char\\_device](#) (void)
- Unregister the DM7820 character device and free the character device major number.*
- static int [dm7820\\_validate\\_device](#) (const [dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device)
- Given what is assumed to be the address of a DM7820 device descriptor, make sure it corresponds to a valid DM7820 device descriptor.*
- static int [dm7820\\_validate\\_pci\\_access](#) (const [dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, const [dm7820\\_pci\\_access\\_request\\_t](#) \*pci\_request)
- Validate a user-space access to one of a device's PCI regions.*
- static int [dm7820\\_write\\_pci\\_region](#) ([dm7820\\_device\\_descriptor\\_t](#) \*dm7820\_device, unsigned long ioctl\_param)
- Write an unsigned value to one of a device's PCI regions.*

## Variables

- static struct file\_operations [dm7820\\_file\\_ops](#)
- File operations supported by driver.*

## 6.27.1 Detailed Description

Definitions for the DM7820 driver.

```
//-----
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
```

```
// Technologies, Inc. Using this software indicates agreement with the
// license terms listed above.
//-----
```

Id:

[dm7820\\_driver.h](#) 86294 2015-03-04 21:36:57Z rgroner

Definition in file [dm7820\\_driver.h](#).

## 6.28 include/dm7820\_globals.h File Reference

Global variables used both in the kernel and in the library.

```
#include "dm7820_registers.h"
#include "dm7820_types.h"
```

### Variables

- static [dm7820\\_interrupt\\_control\\_t dm7820\\_interrupt\\_control](#) []  
*Table of information needed to acknowledge, disable, and enable all interrupt sources.*
- [dm7820\\_minor\\_int\\_reg\\_layout\\_t dm7820\\_minor\\_int\\_reg\\_layout](#) []  
*Table of information providing layout of all minor interrupt registers.*

### 6.28.1 Detailed Description

Global variables used both in the kernel and in the library.

```
@warning
    Only the driver and library source code should include this file.

//-----
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc. Using this software indicates agreement with the
//  license terms listed above.
//-----
```

Id:

[dm7820\\_globals.h](#) 89872 2015-07-08 16:05:17Z rgroner

Definition in file [dm7820\\_globals.h](#).

## 6.29 include/dm7820\_ioctl.h File Reference

Low level ioctl() request descriptor structure and request code definitions.

```
#include <linux/ioctl.h>
#include <linux/types.h>
#include "dm7820_types.h"
```



## Data Structures

- struct [dm7820\\_ioctl\\_region\\_readwrite](#)  
*ioctl() request structure for read from or write to PCI region*
- struct [dm7820\\_ioctl\\_region\\_modify](#)  
*ioctl() request structure for PCI region read/modify/write*
- struct [dm7820\\_ioctl\\_interrupt\\_status](#)  
*ioctl() request structure for getting interrupt status and waiting for an interrupt to occur*
- struct [dm7820\\_dma\\_initialize\\_arguments](#)  
*Arguments for DMA initialization function.*
- union [dm7820\\_dma\\_function\\_arguments](#)  
*Structure encapsulating arguments to all possible DMA functions.*
- struct [dm7820\\_ioctl\\_dma\\_function](#)  
*ioctl() request structure for performing a DMA function*
- union [dm7820\\_ioctl\\_argument](#)  
*ioctl() request structure encapsulating all possible requests. This is what gets passed into the kernel from user space on the ioctl() call.*

## Macros

- #define [DM7820\\_IOCTL\\_MAGIC](#) 'D'  
*Unique 8-bit value used to generate unique ioctl() request codes.*
- #define [DM7820\\_IOCTL\\_REQUEST\\_BASE](#) 0x00  
*First ioctl() request number.*
- #define [DM7820\\_IOCTL\\_REGION\\_READ](#)  
*ioctl() request code for reading from a PCI region*
- #define [DM7820\\_IOCTL\\_REGION\\_WRITE](#)  
*ioctl() request code for writing to a PCI region*
- #define [DM7820\\_IOCTL\\_REGION\\_MODIFY](#)  
*ioctl() request code for PCI region read/modify/write*
- #define [DM7820\\_IOCTL\\_GET\\_INTERRUPT\\_STATUS](#)  
*ioctl() request code for getting interrupt status and waiting for an interrupt to occur*
- #define [DM7820\\_IOCTL\\_DMA\\_FUNCTION](#)  
*ioctl() request code for DMA function*
- #define [DM7820\\_IOCTL\\_WAKEUP](#)  
*ioctl() request code for User ISR thread wake up*
- #define [DM7820\\_IOCTL\\_INTERRUPT\\_INFO](#)  
*ioctl() request code to retrieve interrupt status information*

## Typedefs

- typedef enum  
[dm7820\\_dma\\_manage\\_function](#) [dm7820\\_dma\\_manage\\_function\\_t](#)  
*Functions supported by driver DMA management system.*
- typedef struct  
[dm7820\\_ioctl\\_region\\_readwrite](#) [dm7820\\_ioctl\\_region\\_readwrite\\_t](#)
- typedef struct  
[dm7820\\_ioctl\\_region\\_modify](#) [dm7820\\_ioctl\\_region\\_modify\\_t](#)  
*ioctl() PCI region read/modify/write request descriptor type*
- typedef struct  
[dm7820\\_ioctl\\_interrupt\\_status](#) [dm7820\\_ioctl\\_interrupt\\_status\\_t](#)

- ioctl()* interrupt status request descriptor type
- typedef struct [dm7820\\_dma\\_initialize\\_arguments](#) [dm7820\\_dma\\_initialize\\_arguments\\_t](#)  
*Arguments for DMA initialization function.*
- typedef union [dm7820\\_dma\\_function\\_arguments](#) [dm7820\\_dma\\_function\\_arguments\\_t](#)  
*Structure encapsulating arguments to all possible DMA functions.*
- typedef struct [dm7820\\_ioctl\\_dma\\_function](#) [dm7820\\_ioctl\\_dma\\_function\\_t](#)  
*ioctl() request structure for performing a DMA function*
- typedef union [dm7820\\_ioctl\\_argument](#) [dm7820\\_ioctl\\_argument\\_t](#)  
*ioctl() request descriptor type*

## Enumerations

- enum [dm7820\\_dma\\_manage\\_function](#) {  
[DM7820\\_DMA\\_FUNCTION\\_INITIALIZE](#) = 0, [DM7820\\_DMA\\_FUNCTION\\_STOP](#), [DM7820\\_DMA\\_FUNCTION\\_READ](#), [DM7820\\_DMA\\_FUNCTION\\_WRITE](#),  
[DM7820\\_DMA\\_GET\\_BUFFER\\_ADDR](#) }  
*Functions supported by driver DMA management system.*

### 6.29.1 Detailed Description

Low level `ioctl()` request descriptor structure and request code definitions.

```
//-----
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----
```

Id:

[dm7820\\_ioctl.h](#) 86294 2015-03-04 21:36:57Z rgroner

Definition in file [dm7820\\_ioctl.h](#).

## 6.30 include/dm7820\_library.h File Reference

DM7820 user library definitions.

```
#include <stdint.h>
#include "dm7820_types.h"
#include <pthread.h>
#include <sys/wait.h>
```

## Data Structures

- struct [DM7820\\_Board\\_Descriptor](#)

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

## Macros

- `#define DM7820\_Return\_Status(status, string) if(status != 0) { error(EXIT_FAILURE,errno, "ERROR: %s FAILED\n",string); }`
- `#define DM7820\_DMA\_DEMAND\_OFF\_PCI\_TO\_DM7820 0x00`
- `#define DM7820\_DMA\_DEMAND\_OFF\_DM7820\_TO\_PCI 0x01`
- `#define DM7820\_DMA\_DEMAND\_ON\_PCI\_TO\_DM7820 0x02`
- `#define DM7820\_DMA\_DEMAND\_ON\_DM7820\_TO\_PCI 0x03`
- `#define DM7820\_INCENC\_DISABLE\_PHASE\_FILTER\_TRANSITION(filter, transition) ((filter) |= (1 << (transition)))`  
*Configure an incremental encoder phase filter so that counter value update is disabled for the given transition.*
- `#define DM7820\_INCENC\_ENABLE\_PHASE\_FILTER\_TRANSITION(filter, transition) ((filter) &= ~(1 << (transition)))`  
*Configure an incremental encoder phase filter so that counter value update is enabled for the given transition.*
- `#define DM7820\_INCENC\_RESET\_PHASE\_FILTER(filter) ((filter) = 0x00)`  
*Reset an incremental encoder phase filter.*
- `#define DM7820\_INTERRUPT\_STATUS\_IS\_SOURCE\_PENDING(status, source) (((status) & (0x1LL << (source))) ? 0xFF : 0x00)`  
*Determine whether or not the specified interrupt source is pending in the interrupt status obtained via [DM7820\\_General\\_Get\\_Interrupt\\_Status\(\)](#).*

## Typedefs

- typedef int [DM7820\\_Error](#)  
*DM7820 user library error code type.*
- typedef uint8\_t [dm7820\\_incenc\\_phase\\_filter](#)  
*Incremental encoder phase filter type.*
- typedef struct [DM7820\\_Board\\_Descriptor](#) [DM7820\\_Board\\_Descriptor](#)

## Functions

- [DM7820\\_Error](#) [DM7820\\_AdvInt\\_Get\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, uint8\_t \*occurred)  
*Determine whether or not a status condition has occurred for the given advanced interrupt.*
- [DM7820\\_Error](#) [DM7820\\_AdvInt\\_Read\\_Capture](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [DM7820\\_StdIO\\_Port](#) port, uint16\_t \*value)  
*Read the capture register value for the given advanced interrupt and standard I/O port.*
- [DM7820\\_Error](#) [DM7820\\_AdvInt\\_Set\\_Compare](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [DM7820\\_StdIO\\_Port](#) port, uint16\_t value)  
*Load the compare register for the given advanced interrupt and standard I/O port.*
- [DM7820\\_Error](#) [DM7820\\_AdvInt\\_Set\\_Mask](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [DM7820\\_StdIO\\_Port](#) port, uint16\_t value)  
*Load the mask register for the given advanced interrupt and standard I/O port.*
- [DM7820\\_Error](#) [DM7820\\_AdvInt\\_Set\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [dm7820\\_advint\\_master\\_clock](#) master)  
*Select the master clock for the given advanced interrupt.*

- [DM7820\\_Error DM7820\\_AdvInt\\_Set\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_advint\\_interrupt](#) interrupt, [dm7820\\_advint\\_mode](#) mode)  
*Set the mode for the given advanced interrupt.*
- [DM7820\\_Error DM7820\\_FIFO\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint8\\_t](#) enable)  
*Enable or disable the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Get\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_status\\_condition](#) condition, [uint8\\_t](#) \*occurred)  
*Determine whether or not the specified status condition has occurred for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Initialize](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint32\\_t](#) buffer\_count, [uint32\\_t](#) buffer\_size)  
*Set up direct memory access (DMA) for the given DMA/FIFO channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Create\\_Buffer](#) ([uint16\\_t](#) \*\*buf, [uint32\\_t](#) size)  
*Creates a user space DMA buffer.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Free\\_Buffer](#) ([uint16\\_t](#) \*\*buf, [uint32\\_t](#) size)  
*Frees a previously created user space buffer.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Read](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [void](#) \*user\_buffer, [uint32\\_t](#) num\_bufs)  
*Reads the DMA buffers in the driver.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Write](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [void](#) \*user\_buffer, [uint32\\_t](#) num\_bufs)  
*Copies a user buffer to DMA buffers to be sent into a FIFO.*
- [DM7820\\_Error DM7820\\_Stop\\_DMA](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo)  
*Aborts a DMA transfer on a given channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Configure](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint8\\_t](#) direction, [uint32\\_t](#) transfer\_size)  
*Configure the specified DMA channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_DMA\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint8\\_t](#) enable, [uint8\\_t](#) start)  
*Enable and/or Start a DMA channel.*
- [DM7820\\_Error DM7820\\_FIFO\\_Read](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint16\\_t](#) \*data)  
*Read a single value from the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_DMA\\_Request](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_dma\\_request](#) source)  
*Set DMA request source for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_Data\\_Input](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_data\\_input](#) input)  
*Set data input for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_Input\\_Clock](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_input\\_clock](#) clock)  
*Set input clock for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Set\\_Output\\_Clock](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [dm7820\\_fifo\\_output\\_clock](#) clock)  
*Set output clock for the given FIFO.*
- [DM7820\\_Error DM7820\\_FIFO\\_Write](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_fifo\\_queue](#) fifo, [uint16\\_t](#) data)  
*Write a single value to the given FIFO.*
- [DM7820\\_Error DM7820\\_General\\_Close\\_Board](#) ([DM7820\\_Board\\_Descriptor](#) \*handle)  
*Close a DM7820 device file.*
- [DM7820\\_Error DM7820\\_General\\_Enable\\_Interrupt](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_interrupt\\_source](#) source, [uint8\\_t](#) enable)  
*Enable or disable the given interrupt source.*

- [DM7820\\_Error DM7820\\_General\\_Get\\_Interrupt\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_interrupt\\_info](#) \*interrupt\_info, [uint8\\_t](#) wait\_for\_interrupt)  
*Get a device's interrupt status, optionally waiting for an interrupt to occur.*
- [DM7820\\_Error DM7820\\_General\\_Open\\_Board](#) ([uint8\\_t](#) dev\_num, [DM7820\\_Board\\_Descriptor](#) \*\*handle)  
*Open a DM7820 device file.*
- [DM7820\\_Error DM7820\\_General\\_Get\\_Version\\_Info](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [uint8\\_t](#) \*fpga\_type\_id, [uint8\\_t](#) \*fpga\_version, [uint16\\_t](#) \*svn\_version)  
*Read a device's FPGA and source code revision control versions.*
- [DM7820\\_Error DM7820\\_General\\_Is\\_PCI\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [uint8\\_t](#) \*pci\_master)  
*Determine whether or not a device is PCI master capable.*
- [DM7820\\_Error DM7820\\_General\\_Reset](#) ([DM7820\\_Board\\_Descriptor](#) \*handle)  
*Reset a DM7820 device.*
- [DM7820\\_Error DM7820\\_General\\_RemoveISR](#) ([DM7820\\_Board\\_Descriptor](#) \*handle)  
*Uninstall userspace ISR.*
- [DM7820\\_Error DM7820\\_General\\_StartThread](#) ([int](#)(\*fnct)([void](#) \*), [void](#) \*data)  
*Creates thread to watch for interrupts and call userspace ISR.*
- [void](#) \* [DM7820\\_General\\_WaitForInterrupt](#) ([void](#) \*ptr)  
*Waits for DMA Done interrupts.*
- [DM7820\\_Error DM7820\\_General\\_SetISRPriority](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [int](#) priority)  
*Changes the Priority for the ISR thread.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Configure](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [dm7820\\_incenc\\_phase\\_filter](#) phase\_filter, [dm7820\\_incenc\\_input\\_mode](#) input\_mode, [uint8\\_t](#) enable\_input\_filter, [dm7820\\_incenc\\_channel\\_mode](#) channel\_mode, [uint8\\_t](#) enable\_index)  
*Configure the given incremental encoder.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [uint8\\_t](#) enable)  
*Enable or disable the given incremental encoder.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Enable\\_Hold](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [uint8\\_t](#) enable)  
*Enable or disable value register hold for the given incremental encoder.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Get\\_Independent\\_Value](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [dm7820\\_incenc\\_channel](#) channel, [uint16\\_t](#) \*value)  
*Get 16-bit counter value of the given independent incremental encoder channel.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Get\\_Joined\\_Value](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [uint32\\_t](#) \*value)  
*Get 32-bit counter value of the given independent incremental encoder whose channels are joined.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Get\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [dm7820\\_incenc\\_status\\_condition](#) condition, [uint8\\_t](#) \*occurred)  
*Determine whether or not the specified status condition has occurred for the given incremental encoder.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Set\\_Independent\\_Value](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [dm7820\\_incenc\\_channel](#) channel, [uint16\\_t](#) value)  
*Set 16-bit counter value for the given independent incremental encoder channel.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Set\\_Joined\\_Value](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [uint32\\_t](#) value)  
*Set 32-bit counter value for the given incremental encoder whose channels are joined.*
- [DM7820\\_Error DM7820\\_IncEnc\\_Set\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_incenc\\_encoder](#) encoder, [dm7820\\_incenc\\_master\\_clock](#) master)  
*Set the master clock for the given incremental encoder.*
- [DM7820\\_Error DM7820\\_PWM\\_Enable](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator](#) pwm, [uint8\\_t](#) enable)  
*Enable or disable the given pulse width modulator (PWM).*

- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Period](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [uint32\\_t](#) period)  
*Set the period for the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Period\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [dm7820\\_pwm\\_period\\_master\\_clock](#) master)  
*Set the period master clock for the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Width](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [dm7820\\_pwm\\_output](#) output, [uint16\\_t](#) width)  
*Set the width for the specified output on the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PWM\\_Set\\_Width\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_pwm\\_modulator\\_pwm](#), [dm7820\\_pwm\\_width\\_master\\_clock](#) master)  
*Set the width master clock for the given pulse width modulator (PWM).*
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Master](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_master\\_clock](#) master)  
*Select the master clock for the given programmable clock.*
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_mode](#) mode)  
*Select the mode for the given programmable clock.*
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Period](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [uint32\\_t](#) period)  
*Set the period for the given programmable clock.*
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Start\\_Trigger](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_start\\_trigger](#) start)  
*Set the start trigger for the given programmable clock.*
- [DM7820\\_Error DM7820\\_PrgClk\\_Set\\_Stop\\_Trigger](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_prgclk\\_clock](#) clock, [dm7820\\_prgclk\\_stop\\_trigger](#) stop)  
*Set the stop trigger for the given programmable clock.*
- [DM7820\\_Error DM7820\\_StdIO\\_Get\\_Input](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) \*value)  
*Read a value from the given standard I/O port.*
- [DM7820\\_Error DM7820\\_StdIO\\_Set\\_IO\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) bits, [DM7820\\_StdIO\\_IO\\_Mode](#) mode)  
*Set the mode for specific bits in the given standard I/O port.*
- [DM7820\\_Error DM7820\\_StdIO\\_Set\\_Output](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) value)  
*Write a value to the given standard I/O port.*
- [DM7820\\_Error DM7820\\_StdIO\\_SetPeriph\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Port](#) port, [uint16\\_t](#) bits, [DM7820\\_StdIOPeriph\\_Mode](#) mode)  
*Set the peripheral output mode for specific bits in the given standard I/O port.*
- [DM7820\\_Error DM7820\\_StdIO\\_Strobe\\_Input](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Strobe](#) strobe, [uint8\\_t](#) \*state)  
*Determine state of given strobe signal.*
- [DM7820\\_Error DM7820\\_StdIO\\_Strobe\\_Mode](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Strobe](#) strobe, [uint8\\_t](#) output)  
*Set the direction (input or output) for the given strobe signal.*
- [DM7820\\_Error DM7820\\_StdIO\\_Strobe\\_Output](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [DM7820\\_StdIO\\_Strobe](#) strobe, [uint8\\_t](#) state)  
*Set state of given strobe signal.*
- [DM7820\\_Error DM7820\\_TmrCtr\\_Get\\_Status](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_tmrctr\\_timer](#) timer, [uint8\\_t](#) \*occurred)  
*Determine whether or not a status condition has occurred for the given timer/counter.*
- [DM7820\\_Error DM7820\\_TmrCtr\\_Program](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_tmrctr\\_timer](#) timer, [dm7820\\_tmrctr\\_waveform](#) waveform, [dm7820\\_tmrctr\\_count\\_mode](#) count\_mode, [uint16\\_t](#) divisor)

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

- [DM7820\\_Error DM7820\\_TmrCtr\\_Read](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_tmrctr\\_timer](#) timer, [uint16\\_t](#) \*value)

*Read the value of the given 8254 timer/counter.*

- [DM7820\\_Error DM7820\\_TmrCtr\\_Select\\_Clock](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_tmrctr\\_timer](#) timer, [dm7820\\_tmrctr\\_clock](#) clock)

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

- [DM7820\\_Error DM7820\\_TmrCtr\\_Select\\_Gate](#) ([DM7820\\_Board\\_Descriptor](#) \*handle, [dm7820\\_tmrctr\\_timer](#) timer, [dm7820\\_tmrctr\\_gate](#) gate)

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

### 6.30.1 Detailed Description

DM7820 user library definitions.

```
//-----
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----
```

Id:

[dm7820\\_library.h](#) 86275 2015-03-04 15:53:23Z rgroner

Definition in file [dm7820\\_library.h](#).

## 6.31 include/dm7820\_registers.h File Reference

Register definitions for DM7820 devices.

### Macros

- [#define DM7820\\_BAR0\\_INTCSR](#) 0x68  
*PLX interrupt control/status.*
- [#define DM7820\\_BAR0\\_DMAMODE0](#) 0x80  
*PLX DMA channel 0 mode.*
- [#define DM7820\\_BAR0\\_DMAPADR0](#) 0x84  
*PLX DMA channel 0 PCI address.*
- [#define DM7820\\_BAR0\\_DMALADR0](#) 0x88  
*PLX DMA channel 0 local address.*
- [#define DM7820\\_BAR0\\_DMASIZ0](#) 0x8C  
*PLX DMA channel 0 transfer size.*
- [#define DM7820\\_BAR0\\_DMADPR0](#) 0x90  
*PLX DMA channel 0 descriptor pointer.*
- [#define DM7820\\_BAR0\\_DMAMODE1](#) 0x94



- PLX DMA channel 1 mode.*

  - #define [DM7820\\_BAR0\\_DMAPADR1](#) 0x98

*PLX DMA channel 1 PCI address.*
- #define [DM7820\\_BAR0\\_DMALADR1](#) 0x9C

*PLX DMA channel 1 local address.*
- #define [DM7820\\_BAR0\\_DMASIZ1](#) 0xA0

*PLX DMA channel 1 transfer size.*
- #define [DM7820\\_BAR0\\_DMADPR1](#) 0xA4

*PLX DMA channel 1 descriptor pointer.*
- #define [DM7820\\_BAR0\\_DMACSR0](#) 0xA8

*PLX DMA channel 0 command/status.*
- #define [DM7820\\_BAR0\\_DMACSR1](#) 0xA9

*PLX DMA channel 1 command/status.*
- #define [DM7820\\_BAR0\\_DMAARB](#) 0xAC

*PLX DMA arbitration.*
- #define [DM7820\\_BAR0\\_DMATHR](#) 0xB0

*PLX DMA threshold.*
- #define [DM7820\\_BAR0\\_DMADA0](#) 0xB4

*PLX DMA channel 0 PCI dual address cycles upper address.*
- #define [DM7820\\_BAR0\\_DMADA1](#) 0xB8

*PLX DMA channel 1 PCI dual address cycles upper address.*
- #define [DM7820\\_BAR1\\_INTCSR](#) 0x68

*PLX interrupt control/status.*
- #define [DM7820\\_BAR1\\_DMAMODE0](#) 0x80

*PLX DMA channel 0 mode.*
- #define [DM7820\\_BAR1\\_DMAPADR0](#) 0x84

*PLX DMA channel 0 PCI address.*
- #define [DM7820\\_BAR1\\_DMALADR0](#) 0x88

*PLX DMA channel 0 local address.*
- #define [DM7820\\_BAR1\\_DMASIZ0](#) 0x8C

*PLX DMA channel 0 transfer size.*
- #define [DM7820\\_BAR1\\_DMADPR0](#) 0x90

*PLX DMA channel 0 descriptor pointer.*
- #define [DM7820\\_BAR1\\_DMAMODE1](#) 0x94

*PLX DMA channel 1 mode.*
- #define [DM7820\\_BAR1\\_DMAPADR1](#) 0x98

*PLX DMA channel 1 PCI address.*
- #define [DM7820\\_BAR1\\_DMALADR1](#) 0x9C

*PLX DMA channel 1 local address.*
- #define [DM7820\\_BAR1\\_DMASIZ1](#) 0xA0

*PLX DMA channel 1 transfer size.*
- #define [DM7820\\_BAR1\\_DMADPR1](#) 0xA4

*PLX DMA channel 1 descriptor pointer.*
- #define [DM7820\\_BAR1\\_DMACSR0](#) 0xA8

*PLX DMA channel 0 command/status.*
- #define [DM7820\\_BAR1\\_DMACSR1](#) 0xA9

*PLX DMA channel 1 command/status.*
- #define [DM7820\\_BAR1\\_DMAARB](#) 0xAC

*PLX DMA arbitration.*
- #define [DM7820\\_BAR1\\_DMATHR](#) 0xB0

*PLX DMA threshold.*



- #define [DM7820\\_BAR1\\_DMADA0](#) 0xB4  
*PLX DMA channel 0 PCI dual address cycles upper address.*
- #define [DM7820\\_BAR1\\_DMADA1](#) 0xB8  
*PLX DMA channel 1 PCI dual address cycles upper address.*
- #define [DM7820\\_BAR2\\_FPGA\\_VERSION](#) 0x0000  
*FPGA version and type identifiers.*
- #define [DM7820\\_BAR2\\_SVN\\_VERSION](#) 0x0002  
*FPGA source code revision control version identifier.*
- #define [DM7820\\_BAR2\\_BOARD\\_RESET](#) 0x0004  
*Board reset.*
- #define [DM7820\\_BAR2\\_BRD\\_STAT](#) 0x0008  
*Board status.*
- #define [DM7820\\_BAR2\\_INTERRUPT\\_ENABLE](#) 0x0010  
*Local interrupt enable.*
- #define [DM7820\\_BAR2\\_INTERRUPT\\_STATUS](#) 0x0012  
*Local interrupt status.*
- #define [DM7820\\_BAR2\\_PORT0\\_OUTPUT](#) 0x0040  
*Port 0 output value.*
- #define [DM7820\\_BAR2\\_PORT0\\_INPUT](#) 0x0042  
*Port 0 input value.*
- #define [DM7820\\_BAR2\\_PORT0\\_TRISTATE](#) 0x0044  
*Port 0 direction.*
- #define [DM7820\\_BAR2\\_PORT0\\_MODE](#) 0x0046  
*Port 0 operating mode.*
- #define [DM7820\\_BAR2\\_PORT1\\_OUTPUT](#) 0x0048  
*Port 1 output value.*
- #define [DM7820\\_BAR2\\_PORT1\\_INPUT](#) 0x004A  
*Port 1 input value.*
- #define [DM7820\\_BAR2\\_PORT1\\_TRISTATE](#) 0x004C  
*Port 1 direction.*
- #define [DM7820\\_BAR2\\_PORT1\\_MODE](#) 0x004E  
*Port 1 operating mode.*
- #define [DM7820\\_BAR2\\_PORT2\\_OUTPUT](#) 0x0050  
*Port 2 output value.*
- #define [DM7820\\_BAR2\\_PORT2\\_INPUT](#) 0x0052  
*Port 2 input value.*
- #define [DM7820\\_BAR2\\_PORT2\\_TRISTATE](#) 0x0054  
*Port 2 direction.*
- #define [DM7820\\_BAR2\\_PORT2\\_MODE](#) 0x0056  
*Port 2 operating mode.*
- #define [DM7820\\_BAR2\\_STROBE\\_STATUS](#) 0x0058  
*Port 2 strobe signal status.*
- #define [DM7820\\_BAR2\\_PORT0\\_PERIPH\\_SEL\\_L](#) 0x0060  
*Port 0 bits 0-7 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT0\\_PERIPH\\_SEL\\_H](#) 0x0062  
*Port 0 bits 8-15 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT1\\_PERIPH\\_SEL\\_L](#) 0x0064  
*Port 1 bits 0-7 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT1\\_PERIPH\\_SEL\\_H](#) 0x0066  
*Port 1 bits 8-15 peripheral select.*
- #define [DM7820\\_BAR2\\_PORT2\\_PERIPH\\_SEL\\_L](#) 0x0068

- Port 2 bits 0-7 peripheral select.*
  - #define [DM7820\\_BAR2\\_PORT2\\_PERIPH\\_SEL\\_H](#) 0x006A
- Port 2 bits 8-15 peripheral select.*
  - #define [PLX9056\\_DMA\\_WIDTH\\_MASK](#) 0x00000003
  - #define [PLX9056\\_DMA\\_WIDTH\\_8](#) 0x00000000
  - #define [PLX9056\\_DMA\\_WIDTH\\_16](#) 0x00000001
  - #define [PLX9056\\_DMA\\_WIDTH\\_32](#) 0x00000002
  - #define [PLX9056\\_DMA\\_WAITSTATES\\_MASK](#) 0x0000003c
  - #define [PLX9056\\_DMA\\_READY](#) 0x00000040
  - #define [PLX9056\\_DMA\\_BURST](#) 0x00000080
  - #define [PLX9056\\_DMA\\_LOCAL\\_BURST](#) 0x00000100
  - #define [PLX9056\\_DMA\\_SCATTERGATHER](#) 0x00000200
  - #define [PLX9056\\_DMA\\_DONE\\_INTERRUPT](#) 0x00000400
  - #define [PLX9056\\_DMA\\_LOCAL\\_ADDRESSING\\_MODE](#) 0x00000800
  - #define [PLX9056\\_DMA\\_DEMAND\\_MODE](#) 0x00001000
  - #define [PLX9056\\_DMA\\_MEMWRITE\\_INV](#) 0x00002000
  - #define [PLX9056\\_DMA\\_EOT\\_ENABLE](#) 0x00004000
  - #define [PLX9056\\_DMA\\_FAST\\_SLOW\\_TERM](#) 0x00008000
  - #define [PLX9056\\_DMA\\_CLEAR\\_COUNT](#) 0x00010000
  - #define [PLX9056\\_DMA\\_INTERRUPT\\_SEL](#) 0x00020000
  - #define [PLX9056\\_DMA\\_DAC\\_CHAIN](#) 0x00040000
  - #define [PLX9056\\_DMA\\_EOT\\_END](#) 0x00080000
  - #define [PLX9056\\_DMA\\_RING\\_MODE](#) 0x00100000
  - #define [PLX9056\\_DMA\\_RING\\_CONTROL](#) 0x00200000
  - #define [DM7820\\_BAR2\\_TC\\_ID](#) 0x0080
- 8254 timer/counter block identifier*
  - #define [DM7820\\_BAR2\\_TC\\_INT](#) 0x0082
- 8254 timer/counter interrupt control/status*
  - #define [DM7820\\_BAR2\\_TC\\_A0\\_CONTROL](#) 0x0084
- 8254 timer/counter A0 control*
  - #define [DM7820\\_BAR2\\_TC\\_A1\\_CONTROL](#) 0x0086
- 8254 timer/counter A1 control*
  - #define [DM7820\\_BAR2\\_TC\\_A2\\_CONTROL](#) 0x0088
- 8254 timer/counter A2 control*
  - #define [DM7820\\_BAR2\\_TC\\_B0\\_CONTROL](#) 0x008A
- 8254 timer/counter B0 control*
  - #define [DM7820\\_BAR2\\_TC\\_B1\\_CONTROL](#) 0x008C
- 8254 timer/counter B1 control*
  - #define [DM7820\\_BAR2\\_TC\\_B2\\_CONTROL](#) 0x008E
- 8254 timer/counter B2 control*
  - #define [DM7820\\_BAR2\\_FIFO0\\_ID](#) 0x00C0
- FIFO 0 block identifier.*
  - #define [DM7820\\_BAR2\\_FIFO0\\_INT](#) 0x00C2
- FIFO 0 interrupt control/status.*
  - #define [DM7820\\_BAR2\\_FIFO0\\_IN\\_CLK](#) 0x00C4
- FIFO 0 input clock.*
  - #define [DM7820\\_BAR2\\_FIFO0\\_OUT\\_CLK](#) 0x00C6
- FIFO 0 output clock.*
  - #define [DM7820\\_BAR2\\_FIFO0\\_IN\\_DATA\\_DREQ](#) 0x00C8
- FIFO 0 data input and PLX DMA request source.*
  - #define [DM7820\\_BAR2\\_FIFO0\\_CON\\_STAT](#) 0x00CA
- FIFO 0 control/status.*

- #define [DM7820\\_BAR2\\_FIFO0\\_RW\\_PORT](#) 0x00CC  
*FIFO 0 PCI read/write port.*
- #define [DM7820\\_BAR2\\_FIFO1\\_ID](#) 0x00D0  
*FIFO 1 block identifier.*
- #define [DM7820\\_BAR2\\_FIFO1\\_INT](#) 0x00D2  
*FIFO 1 interrupt control/status.*
- #define [DM7820\\_BAR2\\_FIFO1\\_IN\\_CLK](#) 0x00D4  
*FIFO 1 input clock.*
- #define [DM7820\\_BAR2\\_FIFO1\\_OUT\\_CLK](#) 0x00D6  
*FIFO 1 output clock.*
- #define [DM7820\\_BAR2\\_FIFO1\\_IN\\_DATA\\_DREQ](#) 0x00D8  
*FIFO 1 data input and PLX DMA request source.*
- #define [DM7820\\_BAR2\\_FIFO1\\_CON\\_STAT](#) 0x00DA  
*FIFO 1 control/status.*
- #define [DM7820\\_BAR2\\_FIFO1\\_RW\\_PORT](#) 0x00DC  
*FIFO 1 PCI read/write port.*
- #define [DM7820\\_BAR2\\_PRGCLK0\\_ID](#) 0x0100  
*Programmable clock 0 block identifier.*
- #define [DM7820\\_BAR2\\_PRGCLK0\\_MODE](#) 0x0102  
*Programmable clock 0 operating mode.*
- #define [DM7820\\_BAR2\\_PRGCLK0\\_CLK](#) 0x0104  
*Programmable clock 0 master clock.*
- #define [DM7820\\_BAR2\\_PRGCLK0\\_START\\_STOP](#) 0x0106  
*Programmable clock 0 start/stop triggers.*
- #define [DM7820\\_BAR2\\_PRGCLK0\\_PERIOD](#) 0x0108  
*Programmable clock 0 period.*
- #define [DM7820\\_BAR2\\_PRGCLK1\\_ID](#) 0x0140  
*Programmable clock 1 block identifier.*
- #define [DM7820\\_BAR2\\_PRGCLK1\\_MODE](#) 0x0142  
*Programmable clock 1 operating mode.*
- #define [DM7820\\_BAR2\\_PRGCLK1\\_CLK](#) 0x0144  
*Programmable clock 1 master clock.*
- #define [DM7820\\_BAR2\\_PRGCLK1\\_START\\_STOP](#) 0x0146  
*Programmable clock 1 start/stop triggers.*
- #define [DM7820\\_BAR2\\_PRGCLK1\\_PERIOD](#) 0x0148  
*Programmable clock 1 period.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_ID](#) 0x0180  
*Programmable clock 2 block identifier.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_MODE](#) 0x0182  
*Programmable clock 2 operating mode.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_CLK](#) 0x0184  
*Programmable clock 2 master clock.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_START\\_STOP](#) 0x0186  
*Programmable clock 2 start/stop triggers.*
- #define [DM7820\\_BAR2\\_PRGCLK2\\_PERIOD](#) 0x0188  
*Programmable clock 2 period.*
- #define [DM7820\\_BAR2\\_PRGCLK3\\_ID](#) 0x01C0  
*Programmable clock 3 block identifier.*
- #define [DM7820\\_BAR2\\_PRGCLK3\\_MODE](#) 0x01C2  
*Programmable clock 3 operating mode.*
- #define [DM7820\\_BAR2\\_PRGCLK3\\_CLK](#) 0x01C4

- *Programmable clock 3 master clock.*  
• #define [DM7820\\_BAR2\\_PRGCLK3\\_START\\_STOP](#) 0x01C6
- *Programmable clock 3 start/stop triggers.*  
• #define [DM7820\\_BAR2\\_PRGCLK3\\_PERIOD](#) 0x01C8
- *Programmable clock 3 period.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_ID](#) 0x0200
- *Advanced interrupt 0 block identifier.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_INT\\_MODE](#) 0x0202
- *Advanced interrupt 0 mode.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_CLK](#) 0x0204
- *Advanced interrupt 0 master clock.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT0\\_MASK](#) 0x0208
- *Advanced interrupt 0 standard I/O port 0 mask.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT1\\_MASK](#) 0x020A
- *Advanced interrupt 0 standard I/O port 1 mask.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT2\\_MASK](#) 0x020C
- *Advanced interrupt 0 standard I/O port 2 mask.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT0\\_CMP](#) 0x0210
- *Advanced interrupt 0 standard I/O port 0 event mode compare.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT1\\_CMP](#) 0x0212
- *Advanced interrupt 0 standard I/O port 1 event mode compare.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT2\\_CMP](#) 0x0214
- *Advanced interrupt 0 standard I/O port 2 event mode compare.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT0\\_CAPT](#) 0x0218
- *Advanced interrupt 0 standard I/O port 0 value capture.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT1\\_CAPT](#) 0x021A
- *Advanced interrupt 0 standard I/O port 1 value capture.*  
• #define [DM7820\\_BAR2\\_ADVINT0\\_PORT2\\_CAPT](#) 0x021C
- *Advanced interrupt 0 standard I/O port 2 value capture.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_ID](#) 0x0240
- *Advanced interrupt 1 block identifier.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_INT\\_MODE](#) 0x0242
- *Advanced interrupt 1 mode.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_CLK](#) 0x0244
- *Advanced interrupt 1 master clock.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT0\\_MASK](#) 0x0248
- *Advanced interrupt 1 standard I/O port 0 mask.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT1\\_MASK](#) 0x024A
- *Advanced interrupt 1 standard I/O port 1 mask.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT2\\_MASK](#) 0x024C
- *Advanced interrupt 1 standard I/O port 2 mask.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT0\\_CMP](#) 0x0250
- *Advanced interrupt 1 standard I/O port 0 event mode compare.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT1\\_CMP](#) 0x0252
- *Advanced interrupt 1 standard I/O port 1 event mode compare.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT2\\_CMP](#) 0x0254
- *Advanced interrupt 1 standard I/O port 2 event mode compare.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT0\\_CAPT](#) 0x0258
- *Advanced interrupt 1 standard I/O port 0 value capture.*  
• #define [DM7820\\_BAR2\\_ADVINT1\\_PORT1\\_CAPT](#) 0x025A
- *Advanced interrupt 1 standard I/O port 1 value capture.*

- #define [DM7820\\_BAR2\\_ADVINT1\\_PORT2\\_CAPT](#) 0x025C  
*Advanced interrupt 1 standard I/O port 2 value capture.*
- #define [DM7820\\_BAR2\\_INCENC0\\_ID](#) 0x0280  
*Incremental encoder 0 block identifier.*
- #define [DM7820\\_BAR2\\_INCENC0\\_INT](#) 0x0282  
*Incremental encoder 0 interrupt control/status.*
- #define [DM7820\\_BAR2\\_INCENC0\\_CLOCK](#) 0x0284  
*Incremental encoder 0 master clock.*
- #define [DM7820\\_BAR2\\_INCENC0\\_MODE](#) 0x0286  
*Incremental encoder 0 operating mode.*
- #define [DM7820\\_BAR2\\_INCENC0\\_VALUEA](#) 0x0288  
*Incremental encoder 0 channel A value.*
- #define [DM7820\\_BAR2\\_INCENC0\\_VALUEB](#) 0x028A  
*Incremental encoder 0 channel B value.*
- #define [DM7820\\_BAR2\\_INCENC1\\_ID](#) 0x02C0  
*Incremental encoder 1 block identifier.*
- #define [DM7820\\_BAR2\\_INCENC1\\_INT](#) 0x02C2  
*Incremental encoder 1 interrupt control/status.*
- #define [DM7820\\_BAR2\\_INCENC1\\_CLOCK](#) 0x02C4  
*Incremental encoder 1 master clock.*
- #define [DM7820\\_BAR2\\_INCENC1\\_MODE](#) 0x02C6  
*Incremental encoder 1 operating mode.*
- #define [DM7820\\_BAR2\\_INCENC1\\_VALUEA](#) 0x02C8  
*Incremental encoder 1 channel A value.*
- #define [DM7820\\_BAR2\\_INCENC1\\_VALUEB](#) 0x02CA  
*Incremental encoder 1 channel B value.*
- #define [DM7820\\_BAR2\\_PWM0\\_ID](#) 0x0300  
*Pulse width modulator 0 block identifier.*
- #define [DM7820\\_BAR2\\_PWM0\\_MODE](#) 0x0302  
*Pulse width modulator 0 mode.*
- #define [DM7820\\_BAR2\\_PWM0\\_CLK](#) 0x0304  
*Pulse width modulator 0 period/width clocks.*
- #define [DM7820\\_BAR2\\_PWM0\\_PERIOD](#) 0x0308  
*Pulse width modulator 0 period.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHA](#) 0x0310  
*Pulse width modulator 0 output A width.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHB](#) 0x0314  
*Pulse width modulator 0 output B width.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHC](#) 0x0318  
*Pulse width modulator 0 output C width.*
- #define [DM7820\\_BAR2\\_PWM0\\_WIDTHD](#) 0x031C  
*Pulse width modulator 0 output D width.*
- #define [DM7820\\_BAR2\\_PWM1\\_ID](#) 0x0340  
*Pulse width modulator 1 block identifier.*
- #define [DM7820\\_BAR2\\_PWM1\\_MODE](#) 0x0342  
*Pulse width modulator 1 mode.*
- #define [DM7820\\_BAR2\\_PWM1\\_CLK](#) 0x0344  
*Pulse width modulator 1 period/width clocks.*
- #define [DM7820\\_BAR2\\_PWM1\\_PERIOD](#) 0x0348  
*Pulse width modulator 1 period.*
- #define [DM7820\\_BAR2\\_PWM1\\_WIDTHA](#) 0x0350

- Pulse width modulator 1 output A width.*
- #define [DM7820\\_BAR2\\_PWM1\\_WIDTHB](#) 0x0354
- Pulse width modulator 1 output B width.*
- #define [DM7820\\_BAR2\\_PWM1\\_WIDTHC](#) 0x0358
- Pulse width modulator 1 output C width.*
- #define [DM7820\\_BAR2\\_PWM1\\_WIDTHD](#) 0x035C
- Pulse width modulator 1 output D width.*
- #define [DM7820\\_BAR2\\_TCA\\_COUNTER\\_0](#) 0x1000
- 8254 timer/counter A timer 0 value*
- #define [DM7820\\_BAR2\\_TCA\\_COUNTER\\_1](#) 0x1004
- 8254 timer/counter A timer 1 value*
- #define [DM7820\\_BAR2\\_TCA\\_COUNTER\\_2](#) 0x1008
- 8254 timer/counter A timer 2 value*
- #define [DM7820\\_BAR2\\_TCA\\_CON\\_WORD](#) 0x100C
- 8254 timer/counter A control word*
- #define [DM7820\\_BAR2\\_TCB\\_COUNTER\\_0](#) 0x1010
- 8254 timer/counter B timer 0 value*
- #define [DM7820\\_BAR2\\_TCB\\_COUNTER\\_1](#) 0x1014
- 8254 timer/counter B timer 1 value*
- #define [DM7820\\_BAR2\\_TCB\\_COUNTER\\_2](#) 0x1018
- 8254 timer/counter B timer 2 value*
- #define [DM7820\\_BAR2\\_TCB\\_CON\\_WORD](#) 0x101C
- 8254 timer/counter B control word*
- #define [DM7820\\_BAR0\\_LENGTH](#) 0x200
- Length in bytes of BAR0 (memory-mapped PLX registers)*
- #define [DM7820\\_BAR1\\_LENGTH](#) 0x100
- Length in bytes of BAR1 (I/O-mapped PLX registers)*
- #define [DM7820\\_BAR2\\_LENGTH](#) 0x2000
- Length in bytes of BAR2 (memory-mapped FPGA registers)*
- #define [DM7820\\_ID\\_TIMER\\_COUNTER](#) 0x1001
- 8254 timer/counter block identifier*
- #define [DM7820\\_ID\\_FIFO](#) 0x2011
- FIFO block identifier.*
- #define [DM7820\\_ID\\_PROGRAMMABLE\\_CLOCK](#) 0x1000
- Programmable clock block identifier.*
- #define [DM7820\\_ID\\_ADVANCED\\_INTERRUPT](#) 0x0001
- Advanced interrupt block identifier.*
- #define [DM7820\\_ID\\_INCREMENTAL\\_ENCODER](#) 0x0002
- Incremental encoder block identifier.*
- #define [DM7820\\_ID\\_PULSE\\_WIDTH\\_MODULATOR](#) 0x0003
- Pulse width modulator block identifier.*
- #define [DM7820\\_ID\\_NONE](#) 0x0000
- Empty block identifier.*
- #define [DM7820\\_FIRST\\_ID\\_OFFSET](#) 0x0080
- Offset of first possible block identifier.*
- #define [DM7820\\_LAST\\_ID\\_OFFSET](#) 0x03C0
- Offset of last possible block identifier.*
- #define [DM7820\\_FPGA\\_VERSION\\_TYPE\\_ID\\_MASK](#) 0xFF00
- Bit mask to extract FPGA type identifier.*
- #define [DM7820\\_FPGA\\_VERSION\\_VERSION\\_MASK](#) 0x00FF
- Bit mask to extract FPGA version identifier.*
- #define [DM7820\\_BOARD\\_RESET\\_DO\\_RESET](#) 0xA5A5
- Value to write to cause a board reset.*

### 6.31.1 Detailed Description

Register definitions for DM7820 devices.

```
//-----
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.
//
//  This software package is dual-licensed.  Source code that is compiled for
//  kernel mode execution is licensed under the GNU General Public License
//  version 2.  For a copy of this license, refer to the file
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact
//  the Free Software Foundation.  Source code that is compiled for user mode
//  execution is licensed under the RTD End-User Software License Agreement.
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded
//  Technologies, Inc.  Using this software indicates agreement with the
//  license terms listed above.
//-----
```

Id:

[dm7820\\_registers.h](#) 86275 2015-03-04 15:53:23Z rgroner

Definition in file [dm7820\\_registers.h](#).

## 6.32 include/dm7820\_types.h File Reference

Type definitions used both in kernel and user space.

### Data Structures

- struct [\\_dm7820\\_interrupt\\_info](#)  
*Interrupt source information.*
- struct [dm7820\\_pci\\_access\\_request](#)  
*PCI region access request descriptor. This structure holds information about a request to read data from or write data to one of a device's PCI regions.*
- struct [dm7820\\_interrupt\\_control](#)  
*Structure containing information needed to acknowledge, disable, and enable a particular interrupt source.*
- struct [dm7820\\_minor\\_int\\_reg\\_layout](#)  
*Minor interrupt register bit layout.*

### Typedefs

- typedef enum [dm7820\\_pci\\_region\\_num](#) [dm7820\\_pci\\_region\\_num\\_t](#)  
*Standard PCI region number type.*
- typedef enum [dm7820\\_pci\\_region\\_access\\_size](#) [dm7820\\_pci\\_region\\_access\\_size\\_t](#)  
*Standard PCI region access size type.*
- typedef enum [\\_DM7820\\_StdIO\\_Port](#) [DM7820\\_StdIO\\_Port](#)  
*Standard I/O port type.*
- typedef enum [\\_DM7820\\_StdIO\\_IO\\_Mode](#) [DM7820\\_StdIO\\_IO\\_Mode](#)  
*Standard I/O port mode type.*
- typedef enum [\\_DM7820\\_StdIO\\_Periph\\_Mode](#) [DM7820\\_StdIO\\_Periph\\_Mode](#)  
*Standard I/O port peripheral output mode type.*
- typedef enum [\\_DM7820\\_StdIO\\_Strobe](#) [DM7820\\_StdIO\\_Strobe](#)

- Standard I/O port strobe signal type.*

  - typedef enum [\\_dm7820\\_tmrctr\\_timer](#) dm7820\_tmrctr\_timer
  - 8254 timer/counter type*
  - typedef enum [\\_dm7820\\_tmrctr\\_clock](#) dm7820\_tmrctr\_clock
  - 8254 timer/counter clock selector type*
  - typedef enum [\\_dm7820\\_tmrctr\\_gate](#) dm7820\_tmrctr\_gate
  - 8254 timer/counter gate selector type*
  - typedef enum [\\_dm7820\\_tmrctr\\_waveform](#) dm7820\_tmrctr\_waveform
  - 8254 timer/counter waveform mode selector type*
  - typedef enum [\\_dm7820\\_tmrctr\\_count\\_mode](#) dm7820\_tmrctr\_count\_mode
  - 8254 timer/counter count mode selector type*
  - typedef enum [\\_dm7820\\_prgclk\\_clock](#) dm7820\_prgclk\_clock
  - Programmable clock type.*
  - typedef enum [\\_dm7820\\_prgclk\\_mode](#) dm7820\_prgclk\_mode
  - Programmable clock mode type.*
  - typedef enum [\\_dm7820\\_prgclk\\_master\\_clock](#) dm7820\_prgclk\_master\_clock
  - Programmable clock master clock type.*
  - typedef enum [\\_dm7820\\_prgclk\\_start\\_trigger](#) dm7820\_prgclk\_start\_trigger
  - Programmable clock start trigger type.*
  - typedef enum [\\_dm7820\\_prgclk\\_stop\\_trigger](#) dm7820\_prgclk\_stop\_trigger
  - Programmable clock stop trigger type.*
  - typedef enum [\\_dm7820\\_pwm\\_modulator](#) dm7820\_pwm\_modulator
  - Pulse width modulator type.*
  - typedef enum [\\_dm7820\\_pwm\\_period\\_master\\_clock](#) dm7820\_pwm\_period\_master\_clock
  - Pulse width modulator period master clock type.*
  - typedef enum [\\_dm7820\\_pwm\\_output](#) dm7820\_pwm\_output
  - Pulse width modulator output type.*
  - typedef enum [\\_dm7820\\_pwm\\_width\\_master\\_clock](#) dm7820\_pwm\_width\_master\_clock
  - Pulse width modulator width master clock type.*
  - typedef enum [\\_dm7820\\_incenc\\_encoder](#) dm7820\_incenc\_encoder
  - Incremental encoder type.*
  - typedef enum [\\_dm7820\\_incenc\\_master\\_clock](#) dm7820\_incenc\_master\_clock
  - Incremental encoder master clock type.*
  - typedef enum [\\_dm7820\\_incenc\\_input\\_mode](#) dm7820\_incenc\_input\_mode
  - Incremental encoder input mode type.*
  - typedef enum [\\_dm7820\\_incenc\\_channel\\_mode](#) dm7820\_incenc\_channel\_mode
  - Incremental encoder channel mode type.*
  - typedef enum [\\_dm7820\\_incenc\\_phase\\_transition](#) dm7820\_incenc\_phase\_transition
  - Incremental encoder phase filter transition type.*
  - typedef enum [\\_dm7820\\_incenc\\_channel](#) dm7820\_incenc\_channel
  - Incremental encoder channel type.*



- typedef enum  
[\\_dm7820\\_incenc\\_status\\_condition](#) [dm7820\\_incenc\\_status\\_condition](#)  
*Incremental encoder status condition type.*
- typedef enum [\\_dm7820\\_fifo\\_queue](#) [dm7820\\_fifo\\_queue](#)  
*FIFO type.*
- typedef enum  
[\\_dm7820\\_fifo\\_input\\_clock](#) [dm7820\\_fifo\\_input\\_clock](#)  
*FIFO input clock type.*
- typedef enum  
[\\_dm7820\\_fifo\\_output\\_clock](#) [dm7820\\_fifo\\_output\\_clock](#)  
*FIFO output clock type.*
- typedef enum  
[\\_dm7820\\_fifo\\_dma\\_request](#) [dm7820\\_fifo\\_dma\\_request](#)  
*FIFO DMA request source type.*
- typedef enum  
[\\_dm7820\\_fifo\\_data\\_input](#) [dm7820\\_fifo\\_data\\_input](#)  
*FIFO data input type.*
- typedef enum  
[\\_dm7820\\_fifo\\_status\\_condition](#) [dm7820\\_fifo\\_status\\_condition](#)  
*FIFO status condition type.*
- typedef enum  
[\\_dm7820\\_advint\\_interrupt](#) [dm7820\\_advint\\_interrupt](#)  
*Advanced interrupt type.*
- typedef enum [\\_dm7820\\_advint\\_mode](#) [dm7820\\_advint\\_mode](#)  
*Advanced interrupt mode type.*
- typedef enum  
[\\_dm7820\\_advint\\_master\\_clock](#) [dm7820\\_advint\\_master\\_clock](#)  
*Advanced interrupt master clock type.*
- typedef enum  
[\\_dm7820\\_interrupt\\_source](#) [dm7820\\_interrupt\\_source](#)  
*Interrupt source type.*
- typedef enum  
[\\_dm7820\\_minor\\_interrupt\\_register](#) [dm7820\\_minor\\_interrupt\\_register](#)  
*Minor interrupt control/status register type.*
- typedef struct  
[\\_dm7820\\_interrupt\\_info](#) [dm7820\\_interrupt\\_info](#)  
*Interrupt source information type.*
- typedef struct  
[dm7820\\_pci\\_access\\_request](#) [dm7820\\_pci\\_access\\_request\\_t](#)
- typedef struct  
[dm7820\\_interrupt\\_control](#) [dm7820\\_interrupt\\_control\\_t](#)  
*Interrupt control information type.*
- typedef struct  
[dm7820\\_minor\\_int\\_reg\\_layout](#) [dm7820\\_minor\\_int\\_reg\\_layout\\_t](#)  
*Minor interrupt register bit layout type.*
- typedef uint64\_t [dm7820\\_int\\_source\\_status\\_t](#)  
*Interrupt source status type.*

## Enumerations

- enum `dm7820_pci_region_num` { `DM7820_PCI_REGION_PLX_MEM` = 0, `DM7820_PCI_REGION_PLX_IO`, `DM7820_PCI_REGION_FPGA_MEM` }  
*Standard PCI region number.*
- enum `dm7820_pci_region_access_size` { `DM7820_PCI_REGION_ACCESS_8` = 0, `DM7820_PCI_REGION_ACCESS_16`, `DM7820_PCI_REGION_ACCESS_32` }  
*Desired size in bits of access to standard PCI region.*
- enum `_DM7820_StdIO_Port` { `DM7820_STDIO_PORT_0` = 0, `DM7820_STDIO_PORT_1`, `DM7820_STDIO_PORT_2` }  
*Standard I/O ports.*
- enum `_DM7820_StdIO_IO_Mode` { `DM7820_STDIO_MODE_INPUT` = 0, `DM7820_STDIO_MODE_OUTPUT`, `DM7820_STDIO_MODE_PER_OUT` }  
*Standard I/O port modes.*
- enum `_DM7820_StdIO_Periph_Mode` { `DM7820_STDIO_PERIPH_PWM` = 0x0, `DM7820_STDIO_PERIPH_CLK_OTHER`, `DM7820_STDIO_PERIPH_FIFO_0`, `DM7820_STDIO_PERIPH_FIFO_1` }  
*Standard I/O port peripheral output modes.*
- enum `_DM7820_StdIO_Strobe` { `DM7820_STDIO_STROBE_1` = 0, `DM7820_STDIO_STROBE_2` }  
*Strobe signals.*
- enum `_dm7820_tmrctr_timer` { `DM7820_TMRCTR_TIMER_A_0` = 0, `DM7820_TMRCTR_TIMER_A_1`, `DM7820_TMRCTR_TIMER_A_2`, `DM7820_TMRCTR_TIMER_B_0`, `DM7820_TMRCTR_TIMER_B_1`, `DM7820_TMRCTR_TIMER_B_2` }  
*8254 timers/counters*
- enum `_dm7820_tmrctr_clock` { `DM7820_TMRCTR_CLOCK_5_MHZ` = 0, `DM7820_TMRCTR_CLOCK_RESERVED`, `DM7820_TMRCTR_CLOCK_8254_A_0`, `DM7820_TMRCTR_CLOCK_8254_A_1`, `DM7820_TMRCTR_CLOCK_8254_A_2`, `DM7820_TMRCTR_CLOCK_8254_B_0`, `DM7820_TMRCTR_CLOCK_8254_B_1`, `DM7820_TMRCTR_CLOCK_8254_B_2`, `DM7820_TMRCTR_CLOCK_PROG_CLOCK_0`, `DM7820_TMRCTR_CLOCK_PROG_CLOCK_1`, `DM7820_TMRCTR_CLOCK_PROG_CLOCK_2`, `DM7820_TMRCTR_CLOCK_PROG_CLOCK_3`, `DM7820_TMRCTR_CLOCK_STROBE_1`, `DM7820_TMRCTR_CLOCK_STROBE_2`, `DM7820_TMRCTR_CLOCK_INV_STROBE_1`, `DM7820_TMRCTR_CLOCK_INV_STROBE_2` }  
*8254 timer/counter clock selectors*
- enum `_dm7820_tmrctr_gate` { `DM7820_TMRCTR_GATE_LOGIC_0` = 0, `DM7820_TMRCTR_GATE_LOGIC_1`, `DM7820_TMRCTR_GATE_8254_A_0`, `DM7820_TMRCTR_GATE_8254_A_1`, `DM7820_TMRCTR_GATE_8254_A_2`, `DM7820_TMRCTR_GATE_8254_B_0`, `DM7820_TMRCTR_GATE_8254_B_1`, `DM7820_TMRCTR_GATE_8254_B_2`, `DM7820_TMRCTR_GATE_PROG_CLOCK_0`, `DM7820_TMRCTR_GATE_PROG_CLOCK_1`, `DM7820_TMRCTR_GATE_PROG_CLOCK_2`, `DM7820_TMRCTR_GATE_PROG_CLOCK_3`, `DM7820_TMRCTR_GATE_STROBE_1`, `DM7820_TMRCTR_GATE_STROBE_2`, `DM7820_TMRCTR_GATE_INV_STROBE_1`, `DM7820_TMRCTR_GATE_INV_STROBE_2`, `DM7820_TMRCTR_GATE_PORT_2_BIT_0`, `DM7820_TMRCTR_GATE_PORT_2_BIT_1`, `DM7820_TMRCTR_GATE_PORT_2_BIT_2`, `DM7820_TMRCTR_GATE_PORT_2_BIT_3`, `DM7820_TMRCTR_GATE_PORT_2_BIT_4`, `DM7820_TMRCTR_GATE_PORT_2_BIT_5`, `DM7820_TMRCTR_GATE_PORT_2_BIT_6`, `DM7820_TMRCTR_GATE_PORT_2_BIT_7`, `DM7820_TMRCTR_GATE_PORT_2_BIT_8`, `DM7820_TMRCTR_GATE_PORT_2_BIT_9`, `DM7820_TMRCTR_GATE_PORT_2_BIT_10`, `DM7820_TMRCTR_GATE_PORT_2_BIT_11`, `DM7820_TMRCTR_GATE_PORT_2_BIT_12`, `DM7820_TMRCTR_GATE_PORT_2_BIT_13`, `DM7820_TMRCTR_GATE_PORT_2_BIT_14`, `DM7820_TMRCTR_GATE_PORT_2_BIT_15` }  
*8254 timer/counter gate selectors*
- enum `_dm7820_tmrctr_waveform` { `DM7820_TMRCTR_WAVEFORM_EVENT_CTR` = 0, `DM7820_TMRCTR_WAVEFORM_PROG_ONE_SHOT`, `DM7820_TMRCTR_WAVEFORM_RATE_GENERATOR`, `DM7820_TMRCTR_WAVEFORM_SQUARE-`

```
_WAVE,
DM7820_TMRCTR_WAVEFORM_SOFTWARE_STROBE, DM7820_TMRCTR_WAVEFORM_HARDWAR-
E_STROBE }
```

*8254 timer/counter waveform mode selectors*

- enum `_dm7820_tmrctr_count_mode` { `DM7820_TMRCTR_COUNT_MODE_BINARY` = 0, `DM7820_TMRCTR_COUNT_MODE_BCD` }

*8254 timer/counter count mode selectors*

- enum `_dm7820_prghclk_clock` { `DM7820_PRGCLK_CLOCK_0` = 0, `DM7820_PRGCLK_CLOCK_1`, `DM7820_PRGCLK_CLOCK_2`, `DM7820_PRGCLK_CLOCK_3` }

*Programmable clocks.*

- enum `_dm7820_prghclk_mode` { `DM7820_PRGCLK_MODE_DISABLED` = 0, `DM7820_PRGCLK_MODE_CONTINUOUS`, `DM7820_PRGCLK_MODE_RESERVED`, `DM7820_PRGCLK_MODE_ONE_SHOT` }

*Programmable clock modes.*

- enum `_dm7820_prghclk_master_clock` { `DM7820_PRGCLK_MASTER_25_MHZ` = 0, `DM7820_PRGCLK_MASTER_SAMPLE_CLOCK`, `DM7820_PRGCLK_MASTER_8254_A_0`, `DM7820_PRGCLK_MASTER_8254_A_1`, `DM7820_PRGCLK_MASTER_8254_A_2`, `DM7820_PRGCLK_MASTER_8254_B_0`, `DM7820_PRGCLK_MASTER_8254_B_1`, `DM7820_PRGCLK_MASTER_8254_B_2`, `DM7820_PRGCLK_MASTER_PROG_CLOCK_0`, `DM7820_PRGCLK_MASTER_PROG_CLOCK_1`, `DM7820_PRGCLK_MASTER_PROG_CLOCK_2`, `DM7820_PRGCLK_MASTER_PROG_CLOCK_3`, `DM7820_PRGCLK_MASTER_STROBE_1`, `DM7820_PRGCLK_MASTER_STROBE_2`, `DM7820_PRGCLK_MASTER_INV_STROBE_1`, `DM7820_PRGCLK_MASTER_INV_STROBE_2` }

*Programmable clock master clocks.*

- enum `_dm7820_prghclk_start_trigger` { `DM7820_PRGCLK_START_IMMEDIATE` = 0, `DM7820_PRGCLK_START_RESERVED_1`, `DM7820_PRGCLK_START_8254_A_0`, `DM7820_PRGCLK_START_8254_A_1`, `DM7820_PRGCLK_START_8254_A_2`, `DM7820_PRGCLK_START_8254_B_0`, `DM7820_PRGCLK_START_8254_B_1`, `DM7820_PRGCLK_START_8254_B_2`, `DM7820_PRGCLK_START_PROG_CLOCK_0`, `DM7820_PRGCLK_START_PROG_CLOCK_1`, `DM7820_PRGCLK_START_PROG_CLOCK_2`, `DM7820_PRGCLK_START_PROG_CLOCK_3`, `DM7820_PRGCLK_START_STROBE_1`, `DM7820_PRGCLK_START_STROBE_2`, `DM7820_PRGCLK_START_INV_STROBE_1`, `DM7820_PRGCLK_START_INV_STROBE_2`, `DM7820_PRGCLK_START_ADVANCED_INT_0`, `DM7820_PRGCLK_START_ADVANCED_INT_1`, `DM7820_PRGCLK_START_8254_INT`, `DM7820_PRGCLK_START_RESERVED_2`, `DM7820_PRGCLK_START_INC_ENCODER_0_INT`, `DM7820_PRGCLK_START_INC_ENCODER_1_INT`, `DM7820_PRGCLK_START_RESERVED_3`, `DM7820_PRGCLK_START_RESERVED_4`, `DM7820_PRGCLK_START_PWM_0_INT`, `DM7820_PRGCLK_START_PWM_1_INT`, `DM7820_PRGCLK_START_PROG_CLOCK_0_INT`, `DM7820_PRGCLK_START_PROG_CLOCK_1_INT`, `DM7820_PRGCLK_START_PROG_CLOCK_2_INT`, `DM7820_PRGCLK_START_PROG_CLOCK_3_INT`, `DM7820_PRGCLK_START_FIFO_0_INT`, `DM7820_PRGCLK_START_FIFO_1_INT` }

*Programmable clock start triggers.*

- enum `_dm7820_prghclk_stop_trigger` { `DM7820_PRGCLK_STOP_NONE` = 0, `DM7820_PRGCLK_STOP_RESERVED_1`, `DM7820_PRGCLK_STOP_8254_A_0`, `DM7820_PRGCLK_STOP_8254_A_1`, `DM7820_PRGCLK_STOP_8254_A_2`, `DM7820_PRGCLK_STOP_8254_B_0`, `DM7820_PRGCLK_STOP_8254_B_1`, `DM7820_PRGCLK_STOP_8254_B_2`, `DM7820_PRGCLK_STOP_PROG_CLOCK_0`, `DM7820_PRGCLK_STOP_PROG_CLOCK_1`, `DM7820_PRGCLK_STOP_PROG_CLOCK_2`, `DM7820_PRGCLK_STOP_PROG_CLOCK_3`, `DM7820_PRGCLK_STOP_STROBE_1`, `DM7820_PRGCLK_STOP_STROBE_2`, `DM7820_PRGCLK_STOP_INV_STROBE_1`, `DM7820_PRGCLK_STOP_INV_STROBE_2`, `DM7820_PRGCLK_STOP_ADVANCED_INT_0`, `DM7820_PRGCLK_STOP_ADVANCED_INT_1`, `DM7820_PRGCLK_STOP_8254_INT`, `DM7820_PRGCLK_STOP_RESERVED_2`, `DM7820_PRGCLK_STOP_INC_ENCODER_0_INT`, `DM7820_PRGCLK_STOP_INC_ENCODER_1_INT`, `DM7820_PRGCLK_STOP_RESERVED_3`, `DM7820_PRGCLK_STOP_RESERVED_4`, `DM7820_PRGCLK_STOP_PWM_0_INT`, `DM7820_PRGCLK_STOP_PWM_1_INT`, `DM7820_PRGCLK_STOP`

```
OP_PROG_CLOCK_0_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_1_INT,
DM7820_PRGCLK_STOP_PROG_CLOCK_2_INT, DM7820_PRGCLK_STOP_PROG_CLOCK_3_INT, D-
M7820_PRGCLK_STOP_FIFO_0_INT, DM7820_PRGCLK_STOP_FIFO_1_INT }
```

*Programmable clock stop triggers.*

- enum `_dm7820_pwm_modulator`

*Pulse width modulators.*

- enum `_dm7820_pwm_period_master_clock` {  
`DM7820_PWM_PERIOD_MASTER_25_MHZ = 0`, `DM7820_PWM_PERIOD_MASTER_RESERVED`, `D-`  
`M7820_PWM_PERIOD_MASTER_8254_A_0`, `DM7820_PWM_PERIOD_MASTER_8254_A_1`,  
`DM7820_PWM_PERIOD_MASTER_8254_A_2`, `DM7820_PWM_PERIOD_MASTER_8254_B_0`, `DM7820_`  
`PWM_PERIOD_MASTER_8254_B_1`, `DM7820_PWM_PERIOD_MASTER_8254_B_2`,  
`DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_0`, `DM7820_PWM_PERIOD_MASTER_PROG_CLO-`  
`CK_1`, `DM7820_PWM_PERIOD_MASTER_PROG_CLOCK_2`, `DM7820_PWM_PERIOD_MASTER_PRO-`  
`G_CLOCK_3`,  
`DM7820_PWM_PERIOD_MASTER_STROBE_1`, `DM7820_PWM_PERIOD_MASTER_STROBE_2`, `D-`  
`M7820_PWM_PERIOD_MASTER_INV_STROBE_1`, `DM7820_PWM_PERIOD_MASTER_INV_STROBE_2`  
}

*Pulse width modulator period master clocks.*

- enum `_dm7820_pwm_output` { `DM7820_PWM_OUTPUT_A = 0`, `DM7820_PWM_OUTPUT_B`, `DM7820_P-`  
`WM_OUTPUT_C`, `DM7820_PWM_OUTPUT_D` }

*Pulse width modulator outputs.*

- enum `_dm7820_pwm_width_master_clock` {  
`DM7820_PWM_WIDTH_MASTER_25_MHZ = 0`, `DM7820_PWM_WIDTH_MASTER_RESERVED`, `DM7820-`  
`_PWM_WIDTH_MASTER_8254_A_0`, `DM7820_PWM_WIDTH_MASTER_8254_A_1`,  
`DM7820_PWM_WIDTH_MASTER_8254_A_2`, `DM7820_PWM_WIDTH_MASTER_8254_B_0`, `DM7820_P-`  
`WM_WIDTH_MASTER_8254_B_1`, `DM7820_PWM_WIDTH_MASTER_8254_B_2`,  
`DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_0`, `DM7820_PWM_WIDTH_MASTER_PROG_CLOCK-`  
`_1`, `DM7820_PWM_WIDTH_MASTER_PROG_CLOCK_2`, `DM7820_PWM_WIDTH_MASTER_PROG_CL-`  
`OCK_3`,  
`DM7820_PWM_WIDTH_MASTER_STROBE_1`, `DM7820_PWM_WIDTH_MASTER_STROBE_2`, `DM7820-`  
`_PWM_WIDTH_MASTER_INV_STROBE_1`, `DM7820_PWM_WIDTH_MASTER_INV_STROBE_2` }

*Pulse width modulator width master clocks.*

- enum `_dm7820_incenc_encoder` { `DM7820_INCENC_ENCODER_0 = 0`, `DM7820_INCENC_ENCODER_1`  
}

*Incremental encoders.*

- enum `_dm7820_incenc_master_clock` {  
`DM7820_INCENC_MASTER_25_MHZ = 0`, `DM7820_INCENC_MASTER_RESERVED`, `DM7820_INCENC-`  
`_MASTER_8254_A_0`, `DM7820_INCENC_MASTER_8254_A_1`,  
`DM7820_INCENC_MASTER_8254_A_2`, `DM7820_INCENC_MASTER_8254_B_0`, `DM7820_INCENC_MA-`  
`STER_8254_B_1`, `DM7820_INCENC_MASTER_8254_B_2`,  
`DM7820_INCENC_MASTER_PROG_CLOCK_0`, `DM7820_INCENC_MASTER_PROG_CLOCK_1`, `D-`  
`M7820_INCENC_MASTER_PROG_CLOCK_2`, `DM7820_INCENC_MASTER_PROG_CLOCK_3`,  
`DM7820_INCENC_MASTER_STROBE_1`, `DM7820_INCENC_MASTER_STROBE_2`, `DM7820_INCENC_`  
`MASTER_INV_STROBE_1`, `DM7820_INCENC_MASTER_INV_STROBE_2` }

*Incremental encoder master clocks.*

- enum `_dm7820_incenc_input_mode` { `DM7820_INCENC_INPUT_SINGLE_ENDED = 0`, `DM7820_INCENC-`  
`_INPUT_DIFFERENTIAL` }

*Incremental encoder input modes.*

- enum `_dm7820_incenc_channel_mode` { `DM7820_INCENC_CHANNEL_INDEPENDENT = 0`, `DM7820_IN-`  
`CENC_CHANNEL_JOINED` }

*Incremental encoder channel modes.*

- enum `_dm7820_incenc_phase_transition` {  
`DM7820_INCENC_PHASE_BA_00_TO_01_UP = 0`, `DM7820_INCENC_PHASE_BA_01_TO_11_UP`, `D-`  
`M7820_INCENC_PHASE_BA_11_TO_10_UP`, `DM7820_INCENC_PHASE_BA_10_TO_00_UP`,  
`DM7820_INCENC_PHASE_BA_01_TO_00_DOWN`, `DM7820_INCENC_PHASE_BA_11_TO_01_DOWN`,

```
DM7820_INCENC_PHASE_BA_10_TO_11_DOWN, DM7820_INCENC_PHASE_BA_00_TO_10_DOWN
}
```

*Incremental encoder phase filter transitions.*

- enum `_dm7820_incenc_channel` { `DM7820_INCENC_CHANNEL_A` = 0, `DM7820_INCENC_CHANNEL_B` }

*Incremental encoder channels.*

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

*Incremental encoder status conditions.*

- enum `_dm7820_fifo_queue` { `DM7820_FIFO_QUEUE_0` = 0, `DM7820_FIFO_QUEUE_1` }

*FIFOs.*

- enum `_dm7820_fifo_input_clock` {  
`DM7820_FIFO_INPUT_CLOCK_25_MHZ` = 0, `DM7820_FIFO_INPUT_CLOCK_RESERVED_1`, `DM7820_FIFO_INPUT_CLOCK_8254_A_0`, `DM7820_FIFO_INPUT_CLOCK_8254_A_1`,  
`DM7820_FIFO_INPUT_CLOCK_8254_A_2`, `DM7820_FIFO_INPUT_CLOCK_8254_B_0`, `DM7820_FIFO_INPUT_CLOCK_8254_B_1`, `DM7820_FIFO_INPUT_CLOCK_8254_B_2`,  
`DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0`, `DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1`, `DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2`, `DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3`,  
`DM7820_FIFO_INPUT_CLOCK_STROBE_1`, `DM7820_FIFO_INPUT_CLOCK_STROBE_2`, `DM7820_FIFO_INPUT_CLOCK_INV_STROBE_1`, `DM7820_FIFO_INPUT_CLOCK_INV_STROBE_2`,  
`DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_0`, `DM7820_FIFO_INPUT_CLOCK_ADVANCED_INT_1`, `DM7820_FIFO_INPUT_CLOCK_8254_INT`, `DM7820_FIFO_INPUT_CLOCK_RESERVED_2`,  
`DM7820_FIFO_INPUT_CLOCK_INC_ENCODER_0_INT`, `DM7820_FIFO_INPUT_CLOCK_INC_ENCODER_1_INT`, `DM7820_FIFO_INPUT_CLOCK_RESERVED_3`, `DM7820_FIFO_INPUT_CLOCK_RESERVED_4`,  
`DM7820_FIFO_INPUT_CLOCK_PWM_0_INT`, `DM7820_FIFO_INPUT_CLOCK_PWM_1_INT`, `DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_0_INT`, `DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_1_INT`,  
`DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_2_INT`, `DM7820_FIFO_INPUT_CLOCK_PROG_CLOCK_3_INT`, `DM7820_FIFO_INPUT_CLOCK_PCI_READ`, `DM7820_FIFO_INPUT_CLOCK_PCI_WRITE` }

*FIFO input clocks.*

- enum `_dm7820_fifo_output_clock` {  
`DM7820_FIFO_OUTPUT_CLOCK_25_MHZ` = 0, `DM7820_FIFO_OUTPUT_CLOCK_RESERVED_1`, `DM7820_FIFO_OUTPUT_CLOCK_8254_A_0`, `DM7820_FIFO_OUTPUT_CLOCK_8254_A_1`,  
`DM7820_FIFO_OUTPUT_CLOCK_8254_A_2`, `DM7820_FIFO_OUTPUT_CLOCK_8254_B_0`, `DM7820_FIFO_OUTPUT_CLOCK_8254_B_1`, `DM7820_FIFO_OUTPUT_CLOCK_8254_B_2`,  
`DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3`,  
`DM7820_FIFO_OUTPUT_CLOCK_STROBE_1`, `DM7820_FIFO_OUTPUT_CLOCK_STROBE_2`, `DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_1`, `DM7820_FIFO_OUTPUT_CLOCK_INV_STROBE_2`,  
`DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_0`, `DM7820_FIFO_OUTPUT_CLOCK_ADVANCED_INT_1`, `DM7820_FIFO_OUTPUT_CLOCK_8254_INT`, `DM7820_FIFO_OUTPUT_CLOCK_RESERVED_2`,  
`DM7820_FIFO_OUTPUT_CLOCK_INC_ENCODER_0_INT`, `DM7820_FIFO_OUTPUT_CLOCK_INC_ENCODER_1_INT`, `DM7820_FIFO_OUTPUT_CLOCK_RESERVED_3`, `DM7820_FIFO_OUTPUT_CLOCK_RESERVED_4`,  
`DM7820_FIFO_OUTPUT_CLOCK_PWM_0_INT`, `DM7820_FIFO_OUTPUT_CLOCK_PWM_1_INT`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_0_INT`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_1_INT`,  
`DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_2_INT`, `DM7820_FIFO_OUTPUT_CLOCK_PROG_CLOCK_3_INT`, `DM7820_FIFO_OUTPUT_CLOCK_PCI_READ`, `DM7820_FIFO_OUTPUT_CLOCK_PCI_WRITE` }

*FIFO output clocks.*

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

*FIFO DMA request sources.*

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

*FIFO data inputs.*

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

*FIFO status conditions.*

- enum `_dm7820_advint_interrupt` { `DM7820_ADVINT_INTERRUPT_0 = 0, DM7820_ADVINT_INTERRUPT_1` }

*Advanced interrupts.*

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

*Advanced interrupt modes.*

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

*Advanced interrupt master clocks.*

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

*Interrupt sources.*

- enum `_dm7820_minor_interrupt_register` {  
`DM7820_MINOR_INT_REG_FIFO_0_INT = 0, DM7820_MINOR_INT_REG_FIFO_1_INT, DM7820_MINOR_INT_REG_INCENC_0_INT, DM7820_MINOR_INT_REG_INCENC_1_INT,`  
`DM7820_MINOR_INT_REG_TMRCTR_INT, DM7820_MINOR_INT_REG_NONE` }

*Minor interrupt control/status registers.*

### 6.32.1 Detailed Description

Type definitions used both in kernel and user space.

```
//-----  
//  COPYRIGHT (C) RTD EMBEDDED TECHNOLOGIES, INC.  ALL RIGHTS RESERVED.  
//  
//  This software package is dual-licensed.  Source code that is compiled for  
//  kernel mode execution is licensed under the GNU General Public License  
//  version 2.  For a copy of this license, refer to the file  
//  LICENSE_GPLv2.TXT (which should be included with this software) or contact  
//  the Free Software Foundation.  Source code that is compiled for user mode  
//  execution is licensed under the RTD End-User Software License Agreement.  
//  For a copy of this license, refer to LICENSE.TXT or contact RTD Embedded  
//  Technologies, Inc.  Using this software indicates agreement with the  
//  license terms listed above.  
//-----
```

Id:

[dm7820\\_types.h](#) 86294 2015-03-04 21:36:57Z rgroner

Definition in file [dm7820\\_types.h](#).

# Index

- `_DM7820_StdIO_IO_Mode`
  - DM7820 type standard I/O enumerations, [114](#)
- `_DM7820_StdIO_Periph_Mode`
  - DM7820 type standard I/O enumerations, [114](#)
- `_DM7820_StdIO_Port`
  - DM7820 type standard I/O enumerations, [115](#)
- `_DM7820_StdIO_Strobe`
  - DM7820 type standard I/O enumerations, [115](#)
- `_dm7820_advint_interrupt`
  - DM7820 type advanced interrupt enumerations, [137](#)
- `_dm7820_advint_master_clock`
  - DM7820 type advanced interrupt enumerations, [137](#)
- `_dm7820_advint_mode`
  - DM7820 type advanced interrupt enumerations, [138](#)
- `_dm7820_fifo_data_input`
  - DM7820 type FIFO enumerations, [133](#)
- `_dm7820_fifo_dma_request`
  - DM7820 type FIFO enumerations, [133](#)
- `_dm7820_fifo_input_clock`
  - DM7820 type FIFO enumerations, [134](#)
- `_dm7820_fifo_output_clock`
  - DM7820 type FIFO enumerations, [134](#)
- `_dm7820_fifo_queue`
  - DM7820 type FIFO enumerations, [135](#)
- `_dm7820_fifo_status_condition`
  - DM7820 type FIFO enumerations, [135](#)
- `_dm7820_incenc_channel`
  - DM7820 type incremental encoder enumerations, [129](#)
- `_dm7820_incenc_channel_mode`
  - DM7820 type incremental encoder enumerations, [129](#)
- `_dm7820_incenc_encoder`
  - DM7820 type incremental encoder enumerations, [129](#)
- `_dm7820_incenc_input_mode`
  - DM7820 type incremental encoder enumerations, [129](#)
- `_dm7820_incenc_master_clock`
  - DM7820 type incremental encoder enumerations, [129](#)
- `_dm7820_incenc_phase_transition`
  - DM7820 type incremental encoder enumerations, [130](#)
- `_dm7820_incenc_status_condition`
  - DM7820 type incremental encoder enumerations, [130](#)
- `_dm7820_interrupt_info`
  - error, [145](#)
  - int\_missed, [145](#)
  - int\_remaining, [145](#)
  - source, [145](#)
- `_dm7820_interrupt_source`
  - DM7820 type interrupt enumerations, [140](#)
- `_dm7820_minor_interrupt_register`
  - DM7820 type interrupt enumerations, [141](#)
- `_dm7820_prgclk_clock`
  - DM7820 type programmable clock enumerations, [121](#)
- `_dm7820_prgclk_master_clock`
  - DM7820 type programmable clock enumerations, [121](#)
- `_dm7820_prgclk_mode`
  - DM7820 type programmable clock enumerations, [122](#)
- `_dm7820_prgclk_start_trigger`
  - DM7820 type programmable clock enumerations, [122](#)
- `_dm7820_prgclk_stop_trigger`
  - DM7820 type programmable clock enumerations, [123](#)
- `_dm7820_pwm_output`
  - DM7820 type pulse width modulator enumerations, [126](#)
- `_dm7820_pwm_period_master_clock`
  - DM7820 type pulse width modulator enumerations, [126](#)
- `_dm7820_pwm_width_master_clock`
  - DM7820 type pulse width modulator enumerations, [126](#)
- `_dm7820_tmrctr_clock`
  - DM7820 type timer/counter enumerations, [117](#)
- `_dm7820_tmrctr_count_mode`
  - DM7820 type timer/counter enumerations, [117](#)
- `_dm7820_tmrctr_gate`
  - DM7820 type timer/counter enumerations, [117](#)
- `_dm7820_tmrctr_timer`
  - DM7820 type timer/counter enumerations, [118](#)
- `_dm7820_tmrctr_waveform`
  - DM7820 type timer/counter enumerations, [119](#)
- access
  - `dm7820_ioctl_region_modify`, [157](#)
  - `dm7820_ioctl_region_readwrite`, [159](#)
- access\_sizes



- basic\_test.c, 177
- advint1\_interrupts
  - advint\_event\_interrupt.c, 167
- advint\_event\_interrupt.c
  - advint1\_interrupts, 167
  - ISR, 166
  - main, 166
  - program\_name, 167
- advint\_match\_interrupt.c
  - main, 168
  - program\_name, 169
- advint\_status.c
  - main, 170
  - program\_name, 171
- advint\_strobe\_interrupt.c
  - main, 172
  - program\_name, 173
- allocated
  - dm7820\_pci\_region, 162
- arguments
  - dm7820\_ioctl\_dma\_function, 155
- BUF\_NUM
  - dma\_capture\_buffers.c, 181
  - dma\_capture\_file.c, 184
  - loopback.c, 207
  - pwm\_measure.c, 214
- BUF\_SIZE
  - dma\_capture\_buffers.c, 181
  - dma\_capture\_file.c, 184
  - loopback.c, 207
  - pwm\_measure.c, 214
- bad\_device\_name
  - basic\_test.c, 177
  - library\_test.c, 204
- basic\_test.c
  - INIT\_BAD\_DEVICE\_FILE\_CREATED, 175
  - INIT\_BOARD\_ARRAY\_ALLOCATED, 175
  - INIT\_BOARD\_ARRAY\_OPENED, 175
  - INIT\_NO\_INITIALIZATION, 175
- basic\_test.c
  - access\_sizes, 177
  - bad\_device\_name, 177
  - descriptors, 177
  - device\_count, 177
  - expect\_failure\_and\_check, 176
  - expect\_success, 176
  - init\_state, 177
  - initialization\_state, 175
  - initialization\_state\_t, 175
  - main, 176
  - program\_name, 177
  - region\_names, 178
  - region\_sizes, 178
  - register\_read\_t, 175
  - register\_reads, 178
- board
  - dma\_capture\_buffers.c, 182
  - dma\_capture\_file.c, 186
  - loopback.c, 208
  - pwm\_measure.c, 216
- boards
  - library\_test.c, 205
- brdctl\_device\_info.c
  - main, 179
  - program\_name, 180
- buffer\_address
  - dm7820\_ioctl\_dma\_function, 155
- buffer\_count
  - dm7820\_dma\_initialize\_arguments, 151
- buffer\_size
  - dm7820\_dma\_initialize\_arguments, 151
- bus\_address
  - dm7820\_dma\_descriptor\_t, 150
- DM7820\_FIFO\_INPUT\_CLOCK\_INC\_ENCODER\_0\_I-NT
  - DM7820 type FIFO enumerations, 134
- DM7820\_FIFO\_INPUT\_CLOCK\_INC\_ENCODER\_1\_I-NT
  - DM7820 type FIFO enumerations, 134
- DM7820\_FIFO\_OUTPUT\_CLOCK\_INC\_ENCODER\_0-INT
  - DM7820 type FIFO enumerations, 135
- DM7820\_FIFO\_OUTPUT\_CLOCK\_INC\_ENCODER\_1-INT
  - DM7820 type FIFO enumerations, 135
- DM7820 driver enumerations
  - DM7820\_PCI\_REGION\_ACCESS\_READ, 8
  - DM7820\_PCI\_REGION\_ACCESS\_WRITE, 8
- DM7820 ioctl enumerations
  - DM7820\_DMA\_FUNCTION\_INITIALIZE, 32
  - DM7820\_DMA\_FUNCTION\_READ, 32
  - DM7820\_DMA\_FUNCTION\_STOP, 32
  - DM7820\_DMA\_FUNCTION\_WRITE, 32
  - DM7820\_DMA\_GET\_BUFFER\_ADDR, 32
- DM7820 type advanced interrupt enumerations
  - DM7820\_ADVINT\_INTERRUPT\_0, 137
  - DM7820\_ADVINT\_INTERRUPT\_1, 137
  - DM7820\_ADVINT\_MASTER\_25\_MHZ, 138
  - DM7820\_ADVINT\_MASTER\_8254\_A\_0, 138
  - DM7820\_ADVINT\_MASTER\_8254\_A\_1, 138
  - DM7820\_ADVINT\_MASTER\_8254\_A\_2, 138
  - DM7820\_ADVINT\_MASTER\_8254\_B\_0, 138
  - DM7820\_ADVINT\_MASTER\_8254\_B\_1, 138
  - DM7820\_ADVINT\_MASTER\_8254\_B\_2, 138
  - DM7820\_ADVINT\_MASTER\_INV\_STROBE\_1, 138
  - DM7820\_ADVINT\_MASTER\_INV\_STROBE\_2, 138
  - DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_0, 138
  - DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_1, 138
  - DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_2, 138
  - DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_3, 138

- DM7820\_ADVINT\_MASTER\_RESERVED, [138](#)
- DM7820\_ADVINT\_MASTER\_STROBE\_1, [138](#)
- DM7820\_ADVINT\_MASTER\_STROBE\_2, [138](#)
- DM7820\_ADVINT\_MODE\_DISABLED, [138](#)
- DM7820\_ADVINT\_MODE\_EVENT, [138](#)
- DM7820\_ADVINT\_MODE\_MATCH, [138](#)
- DM7820\_ADVINT\_MODE\_STROBE, [138](#)
- DM7820 type FIFO enumerations
  - DM7280\_FIFO\_INPUT\_CLOCK\_INC\_ENCODER\_0\_INT, [134](#)
  - DM7280\_FIFO\_INPUT\_CLOCK\_INC\_ENCODER\_1\_INT, [134](#)
  - DM7280\_FIFO\_OUTPUT\_CLOCK\_INC\_ENCODER\_0\_INT, [135](#)
  - DM7280\_FIFO\_OUTPUT\_CLOCK\_INC\_ENCODER\_1\_INT, [135](#)
  - DM7820\_FIFO\_0\_DATA\_INPUT\_FIFO\_0\_OUTPUT, [133](#)
  - DM7820\_FIFO\_0\_DATA\_INPUT\_PCI\_DATA, [133](#)
  - DM7820\_FIFO\_0\_DATA\_INPUT\_PORT\_0, [133](#)
  - DM7820\_FIFO\_0\_DATA\_INPUT\_PORT\_2, [133](#)
  - DM7820\_FIFO\_1\_DATA\_INPUT\_INC\_ENCODER\_1\_A, [133](#)
  - DM7820\_FIFO\_1\_DATA\_INPUT\_INC\_ENCODER\_1\_B, [133](#)
  - DM7820\_FIFO\_1\_DATA\_INPUT\_PCI\_DATA, [133](#)
  - DM7820\_FIFO\_1\_DATA\_INPUT\_PORT\_1, [133](#)
  - DM7820\_FIFO\_DMA\_REQUEST\_NOT\_EMPTY, [134](#)
  - DM7820\_FIFO\_DMA\_REQUEST\_NOT\_FULL, [134](#)
  - DM7820\_FIFO\_DMA\_REQUEST\_READ, [134](#)
  - DM7820\_FIFO\_DMA\_REQUEST\_WRITE, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_25\_MHZ, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_0, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_2, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_0, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_2, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_8254\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_ADVANCED\_INT\_0, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_ADVANCED\_INT\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_INV\_STROBE\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_INV\_STROBE\_2, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_READ, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_WRITE, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_1\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_2, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_2\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_3, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_3\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PWM\_0\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_PWM\_1\_INT, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_2, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_3, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_4, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_STROBE\_1, [134](#)
  - DM7820\_FIFO\_INPUT\_CLOCK\_STROBE\_2, [134](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_25\_MHZ, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_0, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_1, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_2, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_0, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_1, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_2, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_INT, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_ADVANCED\_INT\_0, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_ADVANCED\_INT\_1, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_INV\_STROBE\_1, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_INV\_STROBE\_2, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_READ, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_WRITE, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0\_INT, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_1, [135](#)
  - DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_1\_INT, [135](#)

- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_2, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_2\_INT, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_3, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_3\_INT, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PWM\_0\_INT, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PWM\_1\_INT, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_1, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_2, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_3, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_4, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_STROBE\_1, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_STROBE\_2, [135](#)
- DM7820\_FIFO\_QUEUE\_0, [135](#)
- DM7820\_FIFO\_QUEUE\_1, [135](#)
- DM7820\_FIFO\_STATUS\_EMPTY, [136](#)
- DM7820\_FIFO\_STATUS\_FULL, [136](#)
- DM7820\_FIFO\_STATUS\_OVERFLOW, [136](#)
- DM7820\_FIFO\_STATUS\_READ\_REQUEST, [136](#)
- DM7820\_FIFO\_STATUS\_UNDERFLOW, [136](#)
- DM7820\_FIFO\_STATUS\_WRITE\_REQUEST, [136](#)
- DM7820 type incremental encoder enumerations
  - DM7820\_INCENC\_CHANNEL\_A, [129](#)
  - DM7820\_INCENC\_CHANNEL\_B, [129](#)
  - DM7820\_INCENC\_CHANNEL\_INDEPENDENT, [129](#)
  - DM7820\_INCENC\_CHANNEL\_JOINED, [129](#)
  - DM7820\_INCENC\_ENCODER\_0, [129](#)
  - DM7820\_INCENC\_ENCODER\_1, [129](#)
  - DM7820\_INCENC\_INPUT\_DIFFERENTIAL, [129](#)
  - DM7820\_INCENC\_INPUT\_SINGLE\_ENDED, [129](#)
  - DM7820\_INCENC\_MASTER\_25\_MHZ, [130](#)
  - DM7820\_INCENC\_MASTER\_8254\_A\_0, [130](#)
  - DM7820\_INCENC\_MASTER\_8254\_A\_1, [130](#)
  - DM7820\_INCENC\_MASTER\_8254\_A\_2, [130](#)
  - DM7820\_INCENC\_MASTER\_8254\_B\_0, [130](#)
  - DM7820\_INCENC\_MASTER\_8254\_B\_1, [130](#)
  - DM7820\_INCENC\_MASTER\_8254\_B\_2, [130](#)
  - DM7820\_INCENC\_MASTER\_INV\_STROBE\_1, [130](#)
  - DM7820\_INCENC\_MASTER\_INV\_STROBE\_2, [130](#)
  - DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_0, [130](#)
  - DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_1, [130](#)
  - DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_2, [130](#)
  - DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_3, [130](#)
  - DM7820\_INCENC\_MASTER\_RESERVED, [130](#)
  - DM7820\_INCENC\_MASTER\_STROBE\_1, [130](#)
  - DM7820\_INCENC\_MASTER\_STROBE\_2, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_00\_TO\_01\_UP, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_00\_TO\_10\_DOWN, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_01\_TO\_00\_DOWN, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_01\_TO\_11\_UP, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_10\_TO\_00\_UP, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_10\_TO\_11\_DOWN, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_11\_TO\_01\_DOWN, [130](#)
  - DM7820\_INCENC\_PHASE\_BA\_11\_TO\_10\_UP, [130](#)
  - DM7820\_INCENC\_STATUS\_CHANNEL\_A\_NEGATIVE\_ROLLOVER, [131](#)
  - DM7820\_INCENC\_STATUS\_CHANNEL\_A\_POSITIVE\_ROLLOVER, [131](#)
  - DM7820\_INCENC\_STATUS\_CHANNEL\_B\_NEGATIVE\_ROLLOVER, [131](#)
  - DM7820\_INCENC\_STATUS\_CHANNEL\_B\_POSITIVE\_ROLLOVER, [131](#)
- DM7820 type interrupt enumerations
  - DM7820\_INTERRUPT\_ADVINT\_0, [140](#)
  - DM7820\_INTERRUPT\_ADVINT\_1, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_DMA\_DONE, [141](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_EMPTY, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_FULL, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_OVERFLOW, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_READ\_REQUEST, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_UNDERFLOW, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_0\_WRITE\_REQUEST, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_DMA\_DONE, [141](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_EMPTY, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_FULL, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_OVERFLOW, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_READ\_REQUEST, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_UNDERFLOW, [140](#)
  - DM7820\_INTERRUPT\_FIFO\_1\_WRITE\_REQUEST, [140](#)
  - DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_A\_NEGATIVE\_ROLLOVER, [140](#)
  - DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_A\_POSITIVE\_ROLLOVER, [140](#)

- DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_B-\_NEGATIVE\_ROLLOVER, [140](#)
- DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_B-\_POSITIVE\_ROLLOVER, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_A-\_NEGATIVE\_ROLLOVER, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_A-\_POSITIVE\_ROLLOVER, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_B-\_NEGATIVE\_ROLLOVER, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_B-\_POSITIVE\_ROLLOVER, [140](#)
- DM7820\_INTERRUPT\_NONE, [141](#)
- DM7820\_INTERRUPT\_PRGCLK\_0, [140](#)
- DM7820\_INTERRUPT\_PRGCLK\_1, [140](#)
- DM7820\_INTERRUPT\_PRGCLK\_2, [140](#)
- DM7820\_INTERRUPT\_PRGCLK\_3, [140](#)
- DM7820\_INTERRUPT\_PWM\_0, [140](#)
- DM7820\_INTERRUPT\_PWM\_1, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_A\_0, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_A\_1, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_A\_2, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_B\_0, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_B\_1, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_B\_2, [140](#)
- DM7820\_MINOR\_INT\_REG\_FIFO\_0\_INT, [141](#)
- DM7820\_MINOR\_INT\_REG\_FIFO\_1\_INT, [141](#)
- DM7820\_MINOR\_INT\_REG\_INCENC\_0\_INT, [141](#)
- DM7820\_MINOR\_INT\_REG\_INCENC\_1\_INT, [141](#)
- DM7820\_MINOR\_INT\_REG\_NONE, [141](#)
- DM7820\_MINOR\_INT\_REG\_TMRCTR\_INT, [141](#)
- DM7820 type PCI enumerations
  - DM7820\_PCI\_REGION\_ACCESS\_16, [113](#)
  - DM7820\_PCI\_REGION\_ACCESS\_32, [113](#)
  - DM7820\_PCI\_REGION\_ACCESS\_8, [113](#)
  - DM7820\_PCI\_REGION\_FPGA\_MEM, [113](#)
  - DM7820\_PCI\_REGION\_PLX\_IO, [113](#)
  - DM7820\_PCI\_REGION\_PLX\_MEM, [113](#)
- DM7820 type programmable clock enumerations
  - DM7820\_PRGCLK\_CLOCK\_0, [121](#)
  - DM7820\_PRGCLK\_CLOCK\_1, [121](#)
  - DM7820\_PRGCLK\_CLOCK\_2, [121](#)
  - DM7820\_PRGCLK\_CLOCK\_3, [121](#)
  - DM7820\_PRGCLK\_MASTER\_25\_MHZ, [121](#)
  - DM7820\_PRGCLK\_MASTER\_8254\_A\_0, [121](#)
  - DM7820\_PRGCLK\_MASTER\_8254\_A\_1, [121](#)
  - DM7820\_PRGCLK\_MASTER\_8254\_A\_2, [121](#)
  - DM7820\_PRGCLK\_MASTER\_8254\_B\_0, [121](#)
  - DM7820\_PRGCLK\_MASTER\_8254\_B\_1, [121](#)
  - DM7820\_PRGCLK\_MASTER\_8254\_B\_2, [121](#)
  - DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_1, [122](#)
  - DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_2, [122](#)
  - DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_0, [121](#)
  - DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_1, [121](#)
  - DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_2, [122](#)
  - DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_3, [122](#)
  - DM7820\_PRGCLK\_MASTER\_SAMPLE\_CLOCK, [121](#)
  - DM7820\_PRGCLK\_MASTER\_STROBE\_1, [122](#)
  - DM7820\_PRGCLK\_MASTER\_STROBE\_2, [122](#)
  - DM7820\_PRGCLK\_MODE\_CONTINUOUS, [122](#)
  - DM7820\_PRGCLK\_MODE\_DISABLED, [122](#)
  - DM7820\_PRGCLK\_MODE\_ONE\_SHOT, [122](#)
  - DM7820\_PRGCLK\_MODE\_RESERVED, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_A\_0, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_A\_1, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_A\_2, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_B\_0, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_B\_1, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_B\_2, [122](#)
  - DM7820\_PRGCLK\_START\_8254\_INT, [122](#)
  - DM7820\_PRGCLK\_START\_ADVANCED\_INT\_0, [122](#)
  - DM7820\_PRGCLK\_START\_ADVANCED\_INT\_1, [122](#)
  - DM7820\_PRGCLK\_START\_FIFO\_0\_INT, [123](#)
  - DM7820\_PRGCLK\_START\_FIFO\_1\_INT, [123](#)
  - DM7820\_PRGCLK\_START\_IMMEDIATE, [122](#)
  - DM7820\_PRGCLK\_START\_INC\_ENCODER\_0\_I-NT, [122](#)
  - DM7820\_PRGCLK\_START\_INC\_ENCODER\_1\_I-NT, [123](#)
  - DM7820\_PRGCLK\_START\_INV\_STROBE\_1, [122](#)
  - DM7820\_PRGCLK\_START\_INV\_STROBE\_2, [122](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0, [122](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0\_I-NT, [123](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1, [122](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1\_I-NT, [123](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2, [122](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2\_I-NT, [123](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3, [122](#)
  - DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3\_I-NT, [123](#)
  - DM7820\_PRGCLK\_START\_PWM\_0\_INT, [123](#)
  - DM7820\_PRGCLK\_START\_PWM\_1\_INT, [123](#)
  - DM7820\_PRGCLK\_START\_RESERVED\_1, [122](#)
  - DM7820\_PRGCLK\_START\_RESERVED\_2, [122](#)
  - DM7820\_PRGCLK\_START\_RESERVED\_3, [123](#)
  - DM7820\_PRGCLK\_START\_RESERVED\_4, [123](#)
  - DM7820\_PRGCLK\_START\_STROBE\_1, [122](#)
  - DM7820\_PRGCLK\_START\_STROBE\_2, [122](#)
  - DM7820\_PRGCLK\_STOP\_8254\_A\_0, [123](#)
  - DM7820\_PRGCLK\_STOP\_8254\_A\_1, [123](#)

- DM7820\_PRGCLK\_STOP\_8254\_A\_2, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_B\_0, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_B\_1, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_B\_2, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_INT, [123](#)
- DM7820\_PRGCLK\_STOP\_ADVANCED\_INT\_0, [123](#)
- DM7820\_PRGCLK\_STOP\_ADVANCED\_INT\_1, [123](#)
- DM7820\_PRGCLK\_STOP\_FIFO\_0\_INT, [124](#)
- DM7820\_PRGCLK\_STOP\_FIFO\_1\_INT, [124](#)
- DM7820\_PRGCLK\_STOP\_INC\_ENCODER\_0\_INT, [123](#)
- DM7820\_PRGCLK\_STOP\_INC\_ENCODER\_1\_INT, [123](#)
- DM7820\_PRGCLK\_STOP\_INV\_STROBE\_1, [123](#)
- DM7820\_PRGCLK\_STOP\_INV\_STROBE\_2, [123](#)
- DM7820\_PRGCLK\_STOP\_NONE, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_0, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_0\_INT, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_1, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_1\_INT, [124](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_2, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_2\_INT, [124](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_3, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_3\_INT, [124](#)
- DM7820\_PRGCLK\_STOP\_PWM\_0\_INT, [123](#)
- DM7820\_PRGCLK\_STOP\_PWM\_1\_INT, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_1, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_2, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_3, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_4, [123](#)
- DM7820\_PRGCLK\_STOP\_STROBE\_1, [123](#)
- DM7820\_PRGCLK\_STOP\_STROBE\_2, [123](#)
- DM7820 type pulse width modulator enumerations
  - DM7820\_PWM\_OUTPUT\_A, [126](#)
  - DM7820\_PWM\_OUTPUT\_B, [126](#)
  - DM7820\_PWM\_OUTPUT\_C, [126](#)
  - DM7820\_PWM\_OUTPUT\_D, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_25\_MHZ, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_0, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_1, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_2, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_0, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_1, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_2, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_1, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_2, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_0, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_1, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_2, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_3, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_RESERVED, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_STROBE\_1, [126](#)
  - DM7820\_PWM\_PERIOD\_MASTER\_STROBE\_2, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_25\_MHZ, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_0, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_1, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_2, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_0, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_1, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_2, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_1, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_2, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_0, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_1, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_2, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_3, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_RESERVED, [126](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_STROBE\_1, [127](#)
  - DM7820\_PWM\_WIDTH\_MASTER\_STROBE\_2, [127](#)
- DM7820 type standard I/O enumerations
  - DM7820\_STDIO\_MODE\_INPUT, [114](#)
  - DM7820\_STDIO\_MODE\_OUTPUT, [114](#)
  - DM7820\_STDIO\_MODE\_PER\_OUT, [114](#)
  - DM7820\_STDIO\_PERIPH\_CLK\_OTHER, [114](#)
  - DM7820\_STDIO\_PERIPH\_FIFO\_0, [115](#)
  - DM7820\_STDIO\_PERIPH\_FIFO\_1, [115](#)
  - DM7820\_STDIO\_PERIPH\_PWM, [114](#)
  - DM7820\_STDIO\_PORT\_0, [115](#)
  - DM7820\_STDIO\_PORT\_1, [115](#)
  - DM7820\_STDIO\_PORT\_2, [115](#)



- DM7820\_STDIO\_STROBE\_1, [115](#)
- DM7820\_STDIO\_STROBE\_2, [115](#)
- DM7820 type timer/counter enumerations
  - DM7820\_TMRCTR\_CLOCK\_5\_MHZ, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_8254\_A\_0, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_8254\_A\_1, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_8254\_A\_2, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_8254\_B\_0, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_8254\_B\_1, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_8254\_B\_2, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_INV\_STROBE\_1, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_INV\_STROBE\_2, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_0, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_1, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_2, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_3, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_RESERVED, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_STROBE\_1, [117](#)
  - DM7820\_TMRCTR\_CLOCK\_STROBE\_2, [117](#)
  - DM7820\_TMRCTR\_COUNT\_MODE\_BCD, [117](#)
  - DM7820\_TMRCTR\_COUNT\_MODE\_BINARY, [117](#)
  - DM7820\_TMRCTR\_GATE\_8254\_A\_0, [118](#)
  - DM7820\_TMRCTR\_GATE\_8254\_A\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_8254\_A\_2, [118](#)
  - DM7820\_TMRCTR\_GATE\_8254\_B\_0, [118](#)
  - DM7820\_TMRCTR\_GATE\_8254\_B\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_8254\_B\_2, [118](#)
  - DM7820\_TMRCTR\_GATE\_INV\_STROBE\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_INV\_STROBE\_2, [118](#)
  - DM7820\_TMRCTR\_GATE\_LOGIC\_0, [118](#)
  - DM7820\_TMRCTR\_GATE\_LOGIC\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_0, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_10, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_11, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_12, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_13, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_14, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_15, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_2, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_3, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_4, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_5, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_6, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_7, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_8, [118](#)
  - DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_9, [118](#)
  - DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_0, [118](#)
  - DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_2, [118](#)
  - DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_3, [118](#)
  - DM7820\_TMRCTR\_GATE\_STROBE\_1, [118](#)
  - DM7820\_TMRCTR\_GATE\_STROBE\_2, [118](#)
  - DM7820\_TMRCTR\_TIMER\_A\_0, [118](#)
  - DM7820\_TMRCTR\_TIMER\_A\_1, [118](#)
  - DM7820\_TMRCTR\_TIMER\_A\_2, [119](#)
  - DM7820\_TMRCTR\_TIMER\_B\_0, [119](#)
  - DM7820\_TMRCTR\_TIMER\_B\_1, [119](#)
  - DM7820\_TMRCTR\_TIMER\_B\_2, [119](#)
  - DM7820\_TMRCTR\_WAVEFORM\_EVENT\_CTR, [119](#)
  - DM7820\_TMRCTR\_WAVEFORM\_HARDWARE\_STROBE, [119](#)
  - DM7820\_TMRCTR\_WAVEFORM\_PROG\_ONE\_SHOT, [119](#)
  - DM7820\_TMRCTR\_WAVEFORM\_RATE\_GENERATOR, [119](#)
  - DM7820\_TMRCTR\_WAVEFORM\_SOFTWARE\_STROBE, [119](#)
  - DM7820\_TMRCTR\_WAVEFORM\_SQUARE\_WAVE, [119](#)
- DM7820\_ADVINT\_INTERRUPT\_0
  - DM7820 type advanced interrupt enumerations, [137](#)
- DM7820\_ADVINT\_INTERRUPT\_1
  - DM7820 type advanced interrupt enumerations, [137](#)
- DM7820\_ADVINT\_MASTER\_25\_MHZ
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_8254\_A\_0
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_8254\_A\_1
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_8254\_A\_2
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_8254\_B\_0
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_8254\_B\_1
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_8254\_B\_2
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_INV\_STROBE\_1
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_INV\_STROBE\_2
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_0

- DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_1
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_2
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_PROG\_CLOCK\_3
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_RESERVED
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_STROBE\_1
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MASTER\_STROBE\_2
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MODE\_DISABLED
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MODE\_EVENT
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MODE\_MATCH
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_ADVINT\_MODE\_STROBE
  - DM7820 type advanced interrupt enumerations, [138](#)
- DM7820\_DMA\_FUNCTION\_INITIALIZE
  - DM7820 ioctl enumerations, [32](#)
- DM7820\_DMA\_FUNCTION\_READ
  - DM7820 ioctl enumerations, [32](#)
- DM7820\_DMA\_FUNCTION\_STOP
  - DM7820 ioctl enumerations, [32](#)
- DM7820\_DMA\_FUNCTION\_WRITE
  - DM7820 ioctl enumerations, [32](#)
- DM7820\_DMA\_GET\_BUFFER\_ADDR
  - DM7820 ioctl enumerations, [32](#)
- DM7820\_FIFO\_0\_DATA\_INPUT\_FIFO\_0\_OUTPUT
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_0\_DATA\_INPUT\_PCI\_DATA
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_0\_DATA\_INPUT\_PORT\_0
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_0\_DATA\_INPUT\_PORT\_2
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_1\_DATA\_INPUT\_INC\_ENCODER\_1\_A
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_1\_DATA\_INPUT\_INC\_ENCODER\_1\_B
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_1\_DATA\_INPUT\_PCI\_DATA
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_1\_DATA\_INPUT\_PORT\_1
  - DM7820 type FIFO enumerations, [133](#)
- DM7820\_FIFO\_DMA\_REQUEST\_NOT\_EMPTY
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_DMA\_REQUEST\_NOT\_FULL
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_DMA\_REQUEST\_READ
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_DMA\_REQUEST\_WRITE
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_25\_MHZ
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_0
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_A\_2
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_0
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_B\_2
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_8254\_INT
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_ADVANCED\_INT\_0
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_ADVANCED\_INT\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_INV\_STROBE\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_INV\_STROBE\_2
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_READ
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PCI\_WRITE
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_0\_INT
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_1\_INT
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_2
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_2\_INT
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_3
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PROG\_CLOCK\_3\_INT
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_PWM\_0\_INT
  - DM7820 type FIFO enumerations, [134](#)

- DM7820\_FIFO\_INPUT\_CLOCK\_PWM\_1\_INT
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_2
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_3
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_RESERVED\_4
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_STROBE\_1
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_INPUT\_CLOCK\_STROBE\_2
  - DM7820 type FIFO enumerations, [134](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_25\_MHZ
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_0
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_A\_2
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_0
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_B\_2
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_8254\_INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_ADVANCED\_INT\_0
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_ADVANCED\_INT\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_INV\_STROBE\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_INV\_STROBE\_2
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_READ
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PCI\_WRITE
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_0-INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_1-INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_2
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_2-INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_3
  - DM7820 type FIFO enumerations, [135](#)
- DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PROG\_CLOCK\_3-INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PWM\_0\_INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_PWM\_1\_INT
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_2
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_3
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_RESERVED\_4
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_STROBE\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_OUTPUT\_CLOCK\_STROBE\_2
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_QUEUE\_0
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_QUEUE\_1
  - DM7820 type FIFO enumerations, [135](#)
- DM7820\_FIFO\_STATUS\_EMPTY
  - DM7820 type FIFO enumerations, [136](#)
- DM7820\_FIFO\_STATUS\_FULL
  - DM7820 type FIFO enumerations, [136](#)
- DM7820\_FIFO\_STATUS\_OVERFLOW
  - DM7820 type FIFO enumerations, [136](#)
- DM7820\_FIFO\_STATUS\_READ\_REQUEST
  - DM7820 type FIFO enumerations, [136](#)
- DM7820\_FIFO\_STATUS\_UNDERFLOW
  - DM7820 type FIFO enumerations, [136](#)
- DM7820\_FIFO\_STATUS\_WRITE\_REQUEST
  - DM7820 type FIFO enumerations, [136](#)
- DM7820\_INCENC\_CHANNEL\_A
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_CHANNEL\_B
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_CHANNEL\_INDEPENDENT
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_CHANNEL\_JOINED
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_ENCODER\_0
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_ENCODER\_1
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_INPUT\_DIFFERENTIAL
  - DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_INPUT\_SINGLE\_ENDED



- DM7820 type incremental encoder enumerations, [129](#)
- DM7820\_INCENC\_MASTER\_25\_MHZ
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_8254\_A\_0
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_8254\_A\_1
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_8254\_A\_2
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_8254\_B\_0
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_8254\_B\_1
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_8254\_B\_2
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_INV\_STROBE\_1
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_INV\_STROBE\_2
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_0
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_1
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_2
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_PROG\_CLOCK\_3
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_RESERVED
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_STROBE\_1
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_MASTER\_STROBE\_2
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_00\_TO\_01\_UP
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_00\_TO\_10\_DOWN
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_01\_TO\_00\_DOWN
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_01\_TO\_11\_UP
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_10\_TO\_00\_UP
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_10\_TO\_11\_DOWN
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_11\_TO\_01\_DOWN
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_PHASE\_BA\_11\_TO\_10\_UP
  - DM7820 type incremental encoder enumerations, [130](#)
- DM7820\_INCENC\_STATUS\_CHANNEL\_A\_NEGATIVE\_ROLLOVER
  - DM7820 type incremental encoder enumerations, [131](#)
- DM7820\_INCENC\_STATUS\_CHANNEL\_A\_POSITIVE\_ROLLOVER
  - DM7820 type incremental encoder enumerations, [131](#)
- DM7820\_INCENC\_STATUS\_CHANNEL\_B\_NEGATIVE\_ROLLOVER
  - DM7820 type incremental encoder enumerations, [131](#)
- DM7820\_INCENC\_STATUS\_CHANNEL\_B\_POSITIVE\_ROLLOVER
  - DM7820 type incremental encoder enumerations, [131](#)
- DM7820\_INTERRUPT\_ADVINT\_0
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_ADVINT\_1
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_0\_DMA\_DONE
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_INTERRUPT\_FIFO\_0\_EMPTY
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_0\_FULL
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_0\_OVERFLOW
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_0\_READ\_REQUEST
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_0\_UNDERFLOW
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_0\_WRITE\_REQUEST
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_1\_DMA\_DONE
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_INTERRUPT\_FIFO\_1\_EMPTY
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_1\_FULL
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_1\_OVERFLOW
  - DM7820 type interrupt enumerations, [140](#)

- DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_1\_READ\_REQUEST
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_1\_UNDERFLOW
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_FIFO\_1\_WRITE\_REQUEST
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_A\_NEGATIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_A\_POSITIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_B\_NEGATIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_0\_CHANNEL\_B\_POSITIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_A\_NEGATIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_A\_POSITIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_B\_NEGATIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_INCENC\_1\_CHANNEL\_B\_POSITIVE\_ROLLOVER
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_NONE
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_INTERRUPT\_PRGCLK\_0
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_PRGCLK\_1
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_PRGCLK\_2
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_PRGCLK\_3
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_PWM\_0
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_PWM\_1
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_A\_0
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_A\_1
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_A\_2
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_B\_0
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_B\_1
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_INTERRUPT\_TMRCTR\_B\_2
  - DM7820 type interrupt enumerations, [140](#)
- DM7820\_MINOR\_INT\_REG\_FIFO\_0\_INT
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_MINOR\_INT\_REG\_FIFO\_1\_INT
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_MINOR\_INT\_REG\_INCENC\_0\_INT
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_MINOR\_INT\_REG\_INCENC\_1\_INT
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_MINOR\_INT\_REG\_NONE
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_MINOR\_INT\_REG\_TMRCTR\_INT
  - DM7820 type interrupt enumerations, [141](#)
- DM7820\_PCI\_REGION\_ACCESS\_16
  - DM7820 type PCI enumerations, [113](#)
- DM7820\_PCI\_REGION\_ACCESS\_32
  - DM7820 type PCI enumerations, [113](#)
- DM7820\_PCI\_REGION\_ACCESS\_8
  - DM7820 type PCI enumerations, [113](#)
- DM7820\_PCI\_REGION\_ACCESS\_READ
  - DM7820 driver enumerations, [8](#)
- DM7820\_PCI\_REGION\_ACCESS\_WRITE
  - DM7820 driver enumerations, [8](#)
- DM7820\_PCI\_REGION\_FPGA\_MEM
  - DM7820 type PCI enumerations, [113](#)
- DM7820\_PCI\_REGION\_PLX\_IO
  - DM7820 type PCI enumerations, [113](#)
- DM7820\_PCI\_REGION\_PLX\_MEM
  - DM7820 type PCI enumerations, [113](#)
- DM7820\_PRGCLK\_CLOCK\_0
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_CLOCK\_1
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_CLOCK\_2
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_CLOCK\_3
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_25\_MHZ
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_8254\_A\_0
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_8254\_A\_1
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_8254\_A\_2
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_8254\_B\_0
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_8254\_B\_1
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_8254\_B\_2
  - DM7820 type programmable clock enumerations, [121](#)

- DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MASTER\_INV\_STROBE\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_0
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_1
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MASTER\_PROG\_CLOCK\_3
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MASTER\_SAMPLE\_CLOCK
  - DM7820 type programmable clock enumerations, [121](#)
- DM7820\_PRGCLK\_MASTER\_STROBE\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MASTER\_STROBE\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MODE\_CONTINUOUS
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MODE\_DISABLED
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MODE\_ONE\_SHOT
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_MODE\_RESERVED
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_A\_0
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_A\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_A\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_B\_0
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_B\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_B\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_8254\_INT
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_ADVANCED\_INT\_0
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_ADVANCED\_INT\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_FIFO\_0\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_FIFO\_1\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_IMMEDIATE
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_INC\_ENCODER\_0\_INT
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_INC\_ENCODER\_1\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_INV\_STROBE\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_INV\_STROBE\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_0\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_1\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_2\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_PROG\_CLOCK\_3\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_PWM\_0\_INT

- DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_PWM\_1\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_RESERVED\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_RESERVED\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_RESERVED\_3
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_RESERVED\_4
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_START\_STROBE\_1
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_START\_STROBE\_2
  - DM7820 type programmable clock enumerations, [122](#)
- DM7820\_PRGCLK\_STOP\_8254\_A\_0
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_A\_1
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_A\_2
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_B\_0
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_B\_1
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_B\_2
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_8254\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_ADVANCED\_INT\_0
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_ADVANCED\_INT\_1
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_FIFO\_0\_INT
  - DM7820 type programmable clock enumerations, [124](#)
- DM7820\_PRGCLK\_STOP\_FIFO\_1\_INT
  - DM7820 type programmable clock enumerations, [124](#)
- DM7820\_PRGCLK\_STOP\_INC\_ENCODER\_0\_INT
- DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_INC\_ENCODER\_1\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_INV\_STROBE\_1
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_INV\_STROBE\_2
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_NONE
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_0
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_0\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_1
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_1\_INT
  - DM7820 type programmable clock enumerations, [124](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_2
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_2\_INT
  - DM7820 type programmable clock enumerations, [124](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_3
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PROG\_CLOCK\_3\_INT
  - DM7820 type programmable clock enumerations, [124](#)
- DM7820\_PRGCLK\_STOP\_PWM\_0\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_PWM\_1\_INT
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_1
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_2
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_3
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_RESERVED\_4
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_STROBE\_1

- DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PRGCLK\_STOP\_STROBE\_2
  - DM7820 type programmable clock enumerations, [123](#)
- DM7820\_PWM\_OUTPUT\_A
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_OUTPUT\_B
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_OUTPUT\_C
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_OUTPUT\_D
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_25\_MHZ
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_0
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_1
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_8254\_A\_2
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_0
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_1
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_8254\_B\_2
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_1
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_INV\_STROBE\_2
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_0
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_1
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_2
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_PROG\_CLOCK\_3
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_RESERVED
- DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_STROBE\_1
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_PERIOD\_MASTER\_STROBE\_2
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_25\_MHZ
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_0
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_1
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_8254\_A\_2
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_0
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_1
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_8254\_B\_2
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_1
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_INV\_STROBE\_2
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_0
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_1
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_2
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_PROG\_CLOCK\_3
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_RESERVED
  - DM7820 type pulse width modulator enumerations, [126](#)
- DM7820\_PWM\_WIDTH\_MASTER\_STROBE\_1
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_PWM\_WIDTH\_MASTER\_STROBE\_2
  - DM7820 type pulse width modulator enumerations, [127](#)
- DM7820\_STDIO\_MODE\_INPUT
  - DM7820 type standard I/O enumerations, [114](#)



- DM7820\_STDIO\_MODE\_OUTPUT
  - DM7820 type standard I/O enumerations, [114](#)
- DM7820\_STDIO\_MODE\_PER\_OUT
  - DM7820 type standard I/O enumerations, [114](#)
- DM7820\_STDIO\_PERIPH\_CLK\_OTHER
  - DM7820 type standard I/O enumerations, [114](#)
- DM7820\_STDIO\_PERIPH\_FIFO\_0
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_STDIO\_PERIPH\_FIFO\_1
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_STDIO\_PERIPH\_PWM
  - DM7820 type standard I/O enumerations, [114](#)
- DM7820\_STDIO\_PORT\_0
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_STDIO\_PORT\_1
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_STDIO\_PORT\_2
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_STDIO\_STROBE\_1
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_STDIO\_STROBE\_2
  - DM7820 type standard I/O enumerations, [115](#)
- DM7820\_TMRCTR\_CLOCK\_5\_MHZ
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_8254\_A\_0
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_8254\_A\_1
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_8254\_A\_2
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_8254\_B\_0
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_8254\_B\_1
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_8254\_B\_2
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_INV\_STROBE\_1
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_INV\_STROBE\_2
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_0
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_1
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_2
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_PROG\_CLOCK\_3
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_RESERVED
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_STROBE\_1
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_CLOCK\_STROBE\_2
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_COUNT\_MODE\_BCD
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_COUNT\_MODE\_BINARY
  - DM7820 type timer/counter enumerations, [117](#)
- DM7820\_TMRCTR\_GATE\_8254\_A\_0
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_8254\_A\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_8254\_A\_2
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_8254\_B\_0
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_8254\_B\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_8254\_B\_2
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_INV\_STROBE\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_INV\_STROBE\_2
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_LOGIC\_0
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_LOGIC\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_0
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_10
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_11
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_12
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_13
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_14
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_15
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_2
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_3
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_4
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_5
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_6
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_7
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_8
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PORT\_2\_BIT\_9
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_0
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_2
  - DM7820 type timer/counter enumerations, [118](#)

- DM7820\_TMRCTR\_GATE\_PROG\_CLOCK\_3
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_STROBE\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_GATE\_STROBE\_2
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_TIMER\_A\_0
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_TIMER\_A\_1
  - DM7820 type timer/counter enumerations, [118](#)
- DM7820\_TMRCTR\_TIMER\_A\_2
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_TIMER\_B\_0
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_TIMER\_B\_1
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_TIMER\_B\_2
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_WAVEFORM\_EVENT\_CTR
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_WAVEFORM\_HARDWARE\_STR-  
OBE
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_WAVEFORM\_PROG\_ONE\_SHOT
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_WAVEFORM\_RATE\_GENERATO-  
R
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_WAVEFORM\_SOFTWARE\_STR-  
OBE
  - DM7820 type timer/counter enumerations, [119](#)
- DM7820\_TMRCTR\_WAVEFORM\_SQUARE\_WAVE
  - DM7820 type timer/counter enumerations, [119](#)
- DAT\_FILE
  - dma\_capture\_file.c, [184](#)
- DM7820 driver constants, [10](#)
- DM7820 driver enumerations, [8](#)
  - dm7820\_pci\_region\_access\_dir, [8](#)
- DM7820 driver forward declarations, [12](#)
- DM7820 driver functions, [13](#)
  - dm7820\_access\_pci\_region, [15](#)
  - dm7820\_allocate\_irq, [15](#)
  - dm7820\_dequeue\_interrupt, [15](#)
  - dm7820\_disable\_all\_interrupts, [16](#)
  - dm7820\_dma\_check\_xfer, [16](#)
  - dm7820\_dma\_pause, [16](#)
  - dm7820\_dma\_read, [16](#)
  - dm7820\_dma\_stop, [17](#)
  - dm7820\_dma\_write, [17](#)
  - dm7820\_enable\_plx\_interrupts, [17](#)
  - dm7820\_free\_dma\_mappings, [17](#)
  - dm7820\_get\_buffer\_phy\_addr, [18](#)
  - dm7820\_get\_interrupt\_status, [18](#)
  - dm7820\_get\_pci\_master\_status, [18](#)
  - dm7820\_initialize\_device\_descriptor, [19](#)
  - dm7820\_initialize\_dma, [19](#)
  - dm7820\_initialize\_hardware, [20](#)
  - dm7820\_int\_queue\_add, [20](#)
  - dm7820\_interrupt\_handler, [20](#)
  - dm7820\_ioctl, [21](#)
  - dm7820\_load, [21](#)
  - dm7820\_modify\_pci\_region, [21](#)
  - dm7820\_open, [22](#)
  - dm7820\_poll, [22](#)
  - dm7820\_probe\_device\_blocks, [23](#)
  - dm7820\_probe\_devices, [23](#)
  - dm7820\_process\_pci\_regions, [24](#)
  - dm7820\_read\_pci\_region, [25](#)
  - dm7820\_register\_char\_device, [25](#)
  - dm7820\_release, [26](#)
  - dm7820\_release\_resources, [26](#)
  - dm7820\_service\_dma\_function, [26](#)
  - dm7820\_unregister\_char\_device, [27](#)
  - dm7820\_validate\_device, [27](#)
  - dm7820\_validate\_pci\_access, [27](#)
  - dm7820\_write\_pci\_region, [28](#)
- DM7820 driver header file, [7](#)
- DM7820 driver macros, [9](#)
- DM7820 driver structures, [11](#)
- DM7820 global variable header file, [30](#)
- DM7820 ioctl enumerations, [32](#)
  - dm7820\_dma\_manage\_function, [32](#)
- DM7820 ioctl header file, [31](#)
- DM7820 ioctl macros, [34](#)
- DM7820 ioctl structures, [33](#)
  - dm7820\_ioctl\_region\_readwrite\_t, [33](#)
- DM7820 register 8254 timer/counter A offsets, [102](#)
- DM7820 register 8254 timer/counter B offsets, [103](#)
- DM7820 register 8254 timer/counter offsets, [89](#)
- DM7820 register advanced interrupt 0 offsets, [96](#)
- DM7820 register advanced interrupt 1 offsets, [97](#)
- DM7820 register BAR0 (memory-mapped PLX regis-  
ters) offsets, [81](#)
- DM7820 register BAR1 (I/O-mapped PLX registers) off-  
sets, [82](#)
- DM7820 register BAR2 (memory-mapped FPGA regis-  
ters) offsets, [83](#)
- DM7820 register BAR2 Board Reset Register, [110](#)
- DM7820 register BAR2 FPGA Version Register, [109](#)
- DM7820 register BAR2 macros, [108](#)
- DM7820 register board control offsets, [87](#)
- DM7820 register FIFO 0 offsets, [90](#)
- DM7820 register FIFO 1 offsets, [91](#)
- DM7820 register functional block identifier offsets, [107](#)
- DM7820 register functional block identifier values, [106](#)
- DM7820 register functional block identifiers, [105](#)
- DM7820 register header file, [80](#)
- DM7820 register incremental encoder 0 offsets, [98](#)
- DM7820 register incremental encoder 1 offsets, [99](#)
- DM7820 register PCI region lengths, [104](#)
- DM7820 register programmable clock 0 offsets, [92](#)
- DM7820 register programmable clock 1 offsets, [93](#)
- DM7820 register programmable clock 2 offsets, [94](#)
- DM7820 register programmable clock 3 offsets, [95](#)
- DM7820 register pulse width modulator 0 offsets, [100](#)
- DM7820 register pulse width modulator 1 offsets, [101](#)

- DM7820 register standard I/O offsets, [88](#)
- DM7820 type advanced interrupt enumerations, [137](#)
  - [\\_dm7820\\_advint\\_interrupt](#), [137](#)
  - [\\_dm7820\\_advint\\_master\\_clock](#), [137](#)
  - [\\_dm7820\\_advint\\_mode](#), [138](#)
- DM7820 type definition header file, [111](#)
- DM7820 type definition structures, [142](#)
  - [dm7820\\_pci\\_access\\_request\\_t](#), [142](#)
- DM7820 type definition typedefs, [143](#)
- DM7820 type enumerations, [112](#)
- DM7820 type FIFO enumerations, [132](#)
  - [\\_dm7820\\_fifo\\_data\\_input](#), [133](#)
  - [\\_dm7820\\_fifo\\_dma\\_request](#), [133](#)
  - [\\_dm7820\\_fifo\\_input\\_clock](#), [134](#)
  - [\\_dm7820\\_fifo\\_output\\_clock](#), [134](#)
  - [\\_dm7820\\_fifo\\_queue](#), [135](#)
  - [\\_dm7820\\_fifo\\_status\\_condition](#), [135](#)
- DM7820 type incremental encoder enumerations, [128](#)
  - [\\_dm7820\\_incenc\\_channel](#), [129](#)
  - [\\_dm7820\\_incenc\\_channel\\_mode](#), [129](#)
  - [\\_dm7820\\_incenc\\_encoder](#), [129](#)
  - [\\_dm7820\\_incenc\\_input\\_mode](#), [129](#)
  - [\\_dm7820\\_incenc\\_master\\_clock](#), [129](#)
  - [\\_dm7820\\_incenc\\_phase\\_transition](#), [130](#)
  - [\\_dm7820\\_incenc\\_status\\_condition](#), [130](#)
- DM7820 type interrupt enumerations, [139](#)
  - [\\_dm7820\\_interrupt\\_source](#), [140](#)
  - [\\_dm7820\\_minor\\_interrupt\\_register](#), [141](#)
- DM7820 type PCI enumerations, [113](#)
  - [dm7820\\_pci\\_region\\_access\\_size](#), [113](#)
  - [dm7820\\_pci\\_region\\_num](#), [113](#)
- DM7820 type programmable clock enumerations, [120](#)
  - [\\_dm7820\\_prgclk\\_clock](#), [121](#)
  - [\\_dm7820\\_prgclk\\_master\\_clock](#), [121](#)
  - [\\_dm7820\\_prgclk\\_mode](#), [122](#)
  - [\\_dm7820\\_prgclk\\_start\\_trigger](#), [122](#)
  - [\\_dm7820\\_prgclk\\_stop\\_trigger](#), [123](#)
- DM7820 type pulse width modulator enumerations, [125](#)
  - [\\_dm7820\\_pwm\\_output](#), [126](#)
  - [\\_dm7820\\_pwm\\_period\\_master\\_clock](#), [126](#)
  - [\\_dm7820\\_pwm\\_width\\_master\\_clock](#), [126](#)
- DM7820 type standard I/O enumerations, [114](#)
  - [\\_DM7820\\_StdIO\\_IO\\_Mode](#), [114](#)
  - [\\_DM7820\\_StdIO\\_Periph\\_Mode](#), [114](#)
  - [\\_DM7820\\_StdIO\\_Port](#), [115](#)
  - [\\_DM7820\\_StdIO\\_Strobe](#), [115](#)
- DM7820 type timer/counter enumerations, [116](#)
  - [\\_dm7820\\_tmrctr\\_clock](#), [117](#)
  - [\\_dm7820\\_tmrctr\\_count\\_mode](#), [117](#)
  - [\\_dm7820\\_tmrctr\\_gate](#), [117](#)
  - [\\_dm7820\\_tmrctr\\_timer](#), [118](#)
  - [\\_dm7820\\_tmrctr\\_waveform](#), [119](#)
- DM7820 user library 8254 timer/counter functions, [77](#)
  - [DM7820\\_TmrCtr\\_Get\\_Status](#), [77](#)
  - [DM7820\\_TmrCtr\\_Program](#), [78](#)
  - [DM7820\\_TmrCtr\\_Read](#), [78](#)
  - [DM7820\\_TmrCtr\\_Select\\_Clock](#), [78](#)
  - [DM7820\\_TmrCtr\\_Select\\_Gate](#), [79](#)
- DM7820 user library advanced interrupt functions, [42](#)
  - [DM7820\\_AdvInt\\_Get\\_Status](#), [42](#)
  - [DM7820\\_AdvInt\\_Read\\_Capture](#), [43](#)
  - [DM7820\\_AdvInt\\_Set\\_Compare](#), [43](#)
  - [DM7820\\_AdvInt\\_Set\\_Mask](#), [43](#)
  - [DM7820\\_AdvInt\\_Set\\_Master](#), [44](#)
  - [DM7820\\_AdvInt\\_Set\\_Mode](#), [45](#)
- DM7820 user library FIFO functions, [46](#)
  - [DM7820\\_FIFO\\_DMA\\_Configure](#), [47](#)
  - [DM7820\\_FIFO\\_DMA\\_Enable](#), [47](#)
  - [DM7820\\_FIFO\\_DMA\\_Initialize](#), [48](#)
  - [DM7820\\_FIFO\\_DMA\\_Read](#), [49](#)
  - [DM7820\\_FIFO\\_DMA\\_Write](#), [50](#)
  - [DM7820\\_FIFO\\_Enable](#), [50](#)
  - [DM7820\\_FIFO\\_Get\\_Status](#), [51](#)
  - [DM7820\\_FIFO\\_Read](#), [51](#)
  - [DM7820\\_FIFO\\_Write](#), [53](#)
  - [DM7820\\_Stop\\_DMA](#), [54](#)
- DM7820 user library functions, [41](#)
- DM7820 user library general functions, [55](#)
  - [DM7820\\_General\\_Close\\_Board](#), [55](#)
  - [DM7820\\_General\\_Enable\\_Interrupt](#), [56](#)
  - [DM7820\\_General\\_Get\\_Interrupt\\_Status](#), [56](#)
  - [DM7820\\_General\\_Get\\_Version\\_Info](#), [57](#)
  - [DM7820\\_General\\_Is\\_PCI\\_Master](#), [57](#)
  - [DM7820\\_General\\_Open\\_Board](#), [58](#)
  - [DM7820\\_General\\_RemoveISR](#), [58](#)
  - [DM7820\\_General\\_Reset](#), [59](#)
  - [DM7820\\_General\\_SetISRPriority](#), [59](#)
- DM7820 user library header file, [35](#)
- DM7820 user library incremental encoder functions, [60](#)
  - [DM7820\\_IncEnc\\_Configure](#), [60](#)
  - [DM7820\\_IncEnc\\_Enable](#), [61](#)
  - [DM7820\\_IncEnc\\_Enable\\_Hold](#), [61](#)
  - [DM7820\\_IncEnc\\_Get\\_Independent\\_Value](#), [62](#)
  - [DM7820\\_IncEnc\\_Get\\_Joined\\_Value](#), [62](#)
  - [DM7820\\_IncEnc\\_Get\\_Status](#), [63](#)
  - [DM7820\\_IncEnc\\_Set\\_Independent\\_Value](#), [63](#)
  - [DM7820\\_IncEnc\\_Set\\_Joined\\_Value](#), [64](#)
  - [DM7820\\_IncEnc\\_Set\\_Master](#), [64](#)
- DM7820 user library macros, [36](#)
  - [DM7820\\_Return\\_Status](#), [38](#)
- DM7820 user library programmable clock functions, [70](#)
  - [DM7820\\_PrgClk\\_Set\\_Master](#), [70](#)
  - [DM7820\\_PrgClk\\_Set\\_Mode](#), [70](#)
  - [DM7820\\_PrgClk\\_Set\\_Period](#), [71](#)
  - [DM7820\\_PrgClk\\_Set\\_Start\\_Trigger](#), [71](#)
  - [DM7820\\_PrgClk\\_Set\\_Stop\\_Trigger](#), [72](#)
- DM7820 user library pulse width modulator functions, [66](#)
  - [DM7820\\_PWM\\_Enable](#), [66](#)
  - [DM7820\\_PWM\\_Set\\_Period](#), [66](#)
  - [DM7820\\_PWM\\_Set\\_Period\\_Master](#), [67](#)
  - [DM7820\\_PWM\\_Set\\_Width](#), [68](#)
  - [DM7820\\_PWM\\_Set\\_Width\\_Master](#), [68](#)
- DM7820 user library standard I/O functions, [73](#)
  - [DM7820\\_StdIO\\_Get\\_Input](#), [73](#)
  - [DM7820\\_StdIO\\_Set\\_IO\\_Mode](#), [74](#)
  - [DM7820\\_StdIO\\_Set\\_Output](#), [74](#)



- DM7820\_StdIO\_Set\_Periph\_Mode, [74](#)
- DM7820\_StdIO\_Strobe\_Input, [75](#)
- DM7820\_StdIO\_Strobe\_Mode, [75](#)
- DM7820\_StdIO\_Strobe\_Output, [76](#)
- DM7820 user library structures, [40](#)
  - DM7820\_Board\_Descriptor, [40](#)
- DM7820 user library type definitions, [39](#)
- DM7820\_AdvInt\_Get\_Status
  - DM7820 user library advanced interrupt functions, [42](#)
- DM7820\_AdvInt\_Read\_Capture
  - DM7820 user library advanced interrupt functions, [43](#)
- DM7820\_AdvInt\_Set\_Compare
  - DM7820 user library advanced interrupt functions, [43](#)
- DM7820\_AdvInt\_Set\_Mask
  - DM7820 user library advanced interrupt functions, [43](#)
- DM7820\_AdvInt\_Set\_Master
  - DM7820 user library advanced interrupt functions, [44](#)
- DM7820\_AdvInt\_Set\_Mode
  - DM7820 user library advanced interrupt functions, [45](#)
- DM7820\_Board\_Descriptor, [146](#)
  - DM7820 user library structures, [40](#)
  - file\_descriptor, [146](#)
  - isr, [146](#)
  - pid, [146](#)
- DM7820\_FIFO\_DMA\_Configure
  - DM7820 user library FIFO functions, [47](#)
- DM7820\_FIFO\_DMA\_Enable
  - DM7820 user library FIFO functions, [47](#)
- DM7820\_FIFO\_DMA\_Initialize
  - DM7820 user library FIFO functions, [48](#)
- DM7820\_FIFO\_DMA\_Read
  - DM7820 user library FIFO functions, [49](#)
- DM7820\_FIFO\_DMA\_Write
  - DM7820 user library FIFO functions, [50](#)
- DM7820\_FIFO\_Enable
  - DM7820 user library FIFO functions, [50](#)
- DM7820\_FIFO\_Get\_Status
  - DM7820 user library FIFO functions, [51](#)
- DM7820\_FIFO\_Read
  - DM7820 user library FIFO functions, [51](#)
- DM7820\_FIFO\_Set\_Data\_Input
  - DM7820 user library FIFO functions, [52](#)
- DM7820\_FIFO\_Set\_Input\_Clock
  - DM7820 user library FIFO functions, [53](#)
- DM7820\_FIFO\_Set\_Output\_Clock
  - DM7820 user library FIFO functions, [53](#)
- DM7820\_FIFO\_Write
  - DM7820 user library FIFO functions, [53](#)
- DM7820\_General\_Close\_Board
  - DM7820 user library general functions, [55](#)
- DM7820\_General\_Enable\_Interrupt
  - DM7820 user library general functions, [56](#)
- DM7820\_General\_Get\_Interrupt\_Status
  - DM7820 user library general functions, [56](#)
- DM7820\_General\_Get\_Version\_Info
  - DM7820 user library general functions, [57](#)
- DM7820\_General\_Is\_PCI\_Master
  - DM7820 user library general functions, [57](#)
- DM7820\_General\_Open\_Board
  - DM7820 user library general functions, [58](#)
- DM7820\_General\_RemoveISR
  - DM7820 user library general functions, [58](#)
- DM7820\_General\_Reset
  - DM7820 user library general functions, [59](#)
- DM7820\_General\_SetISRPriority
  - DM7820 user library general functions, [59](#)
- DM7820\_IncEnc\_Configure
  - DM7820 user library incremental encoder functions, [60](#)
- DM7820\_IncEnc\_Enable
  - DM7820 user library incremental encoder functions, [61](#)
- DM7820\_IncEnc\_Enable\_Hold
  - DM7820 user library incremental encoder functions, [61](#)
- DM7820\_IncEnc\_Get\_Independent\_Value
  - DM7820 user library incremental encoder functions, [62](#)
- DM7820\_IncEnc\_Get\_Joined\_Value
  - DM7820 user library incremental encoder functions, [62](#)
- DM7820\_IncEnc\_Get\_Status
  - DM7820 user library incremental encoder functions, [63](#)
- DM7820\_IncEnc\_Set\_Independent\_Value
  - DM7820 user library incremental encoder functions, [63](#)
- DM7820\_IncEnc\_Set\_Joined\_Value
  - DM7820 user library incremental encoder functions, [64](#)
- DM7820\_IncEnc\_Set\_Master
  - DM7820 user library incremental encoder functions, [64](#)
- DM7820\_PWM\_Enable
  - DM7820 user library pulse width modulator functions, [66](#)
- DM7820\_PWM\_Set\_Period
  - DM7820 user library pulse width modulator functions, [66](#)
- DM7820\_PWM\_Set\_Period\_Master
  - DM7820 user library pulse width modulator functions, [67](#)
- DM7820\_PWM\_Set\_Width
  - DM7820 user library pulse width modulator functions, [68](#)
- DM7820\_PWM\_Set\_Width\_Master
  - DM7820 user library pulse width modulator functions, [68](#)
- DM7820\_PrgCik\_Set\_Master

- DM7820 user library programmable clock functions, [70](#)
- DM7820\_PrgClk\_Set\_Mode
  - DM7820 user library programmable clock functions, [70](#)
- DM7820\_PrgClk\_Set\_Period
  - DM7820 user library programmable clock functions, [71](#)
- DM7820\_PrgClk\_Set\_Start\_Trigger
  - DM7820 user library programmable clock functions, [71](#)
- DM7820\_PrgClk\_Set\_Stop\_Trigger
  - DM7820 user library programmable clock functions, [72](#)
- DM7820\_Return\_Status
  - DM7820 user library macros, [38](#)
- DM7820\_StdIO\_Get\_Input
  - DM7820 user library standard I/O functions, [73](#)
- DM7820\_StdIO\_Set\_IO\_Mode
  - DM7820 user library standard I/O functions, [74](#)
- DM7820\_StdIO\_Set\_Output
  - DM7820 user library standard I/O functions, [74](#)
- DM7820\_StdIO\_Set\_Periph\_Mode
  - DM7820 user library standard I/O functions, [74](#)
- DM7820\_StdIO\_Strobe\_Input
  - DM7820 user library standard I/O functions, [75](#)
- DM7820\_StdIO\_Strobe\_Mode
  - DM7820 user library standard I/O functions, [75](#)
- DM7820\_StdIO\_Strobe\_Output
  - DM7820 user library standard I/O functions, [76](#)
- DM7820\_Stop\_DMA
  - DM7820 user library FIFO functions, [54](#)
- DM7820\_TmrCtr\_Get\_Status
  - DM7820 user library 8254 timer/counter functions, [77](#)
- DM7820\_TmrCtr\_Program
  - DM7820 user library 8254 timer/counter functions, [78](#)
- DM7820\_TmrCtr\_Read
  - DM7820 user library 8254 timer/counter functions, [78](#)
- DM7820\_TmrCtr\_Select\_Clock
  - DM7820 user library 8254 timer/counter functions, [78](#)
- DM7820\_TmrCtr\_Select\_Gate
  - DM7820 user library 8254 timer/counter functions, [79](#)
- DMA\_BUF\_SIZE
  - [dma\\_capture\\_buffers.c](#), [181](#)
  - [dma\\_capture\\_file.c](#), [185](#)
  - [loopback.c](#), [207](#)
  - [pwm\\_measure.c](#), [215](#)
- DMA\_Transfer\_Size
  - [dm7820\\_ioctl\\_dma\\_function](#), [156](#)
- data
  - [dm7820\\_pci\\_access\\_request](#), [160](#)
- data16
  - [dm7820\\_pci\\_access\\_request](#), [160](#)
- data32
  - [dm7820\\_pci\\_access\\_request](#), [160](#)
- data8
  - [dm7820\\_pci\\_access\\_request](#), [161](#)
- descriptors
  - [basic\\_test.c](#), [177](#)
- device\_count
  - [basic\\_test.c](#), [177](#)
  - [library\\_test.c](#), [205](#)
- device\_lock
  - [dm7820\\_device\\_descriptor](#), [147](#)
- device\_name
  - [dm7820\\_device\\_descriptor](#), [147](#)
- direction
  - [dm7820\\_ioctl\\_dma\\_function](#), [155](#)
- disable\_timers
  - [tmrctr\\_status.c](#), [227](#)
- dm7820\_access\_pci\_region
  - DM7820 driver functions, [15](#)
- dm7820\_allocate\_irq
  - DM7820 driver functions, [15](#)
- dm7820\_dequeue\_interrupt
  - DM7820 driver functions, [15](#)
- dm7820\_device\_descriptor, [147](#)
  - [device\\_lock](#), [147](#)
  - [device\\_name](#), [147](#)
  - [dma\\_buffers\\_post\\_transfer](#), [147](#)
  - [dma\\_buffers\\_pre\\_transfer](#), [147](#)
  - [dma\\_in\\_read\\_direction](#), [148](#)
  - [dma\\_initialized](#), [148](#)
  - [dma\\_size](#), [148](#)
  - [dma\\_wait\\_queue](#), [148](#)
  - [int\\_queue\\_count](#), [148](#)
  - [int\\_queue\\_in](#), [148](#)
  - [int\\_queue\\_missed](#), [148](#)
  - [int\\_queue\\_out](#), [148](#)
  - [int\\_source\\_status](#), [148](#)
  - [int\\_status](#), [149](#)
  - [int\\_wait\\_queue](#), [149](#)
  - [interrupt\\_occurred](#), [149](#)
  - [irq\\_number](#), [149](#)
  - [pci](#), [149](#)
  - [reference\\_count](#), [149](#)
  - [remove\\_isr\\_flag](#), [149](#)
- dm7820\_disable\_all\_interrupts
  - DM7820 driver functions, [16](#)
- dm7820\_dma\_check\_xfer
  - DM7820 driver functions, [16](#)
- dm7820\_dma\_descriptor\_t, [149](#)
  - [bus\\_address](#), [150](#)
  - [virtual\\_address](#), [150](#)
- dm7820\_dma\_function\_arguments, [150](#)
  - [dma\\_init](#), [150](#)
- dm7820\_dma\_initialize\_arguments, [151](#)
  - [buffer\\_count](#), [151](#)
  - [buffer\\_size](#), [151](#)
- dm7820\_dma\_list\_item\_t, [151](#)
  - [dma\\_buffer](#), [152](#)

- list, [152](#)
- dm7820\_dma\_manage\_function
  - DM7820 ioctl enumerations, [32](#)
- dm7820\_dma\_pause
  - DM7820 driver functions, [16](#)
- dm7820\_dma\_read
  - DM7820 driver functions, [16](#)
- dm7820\_dma\_stop
  - DM7820 driver functions, [17](#)
- dm7820\_dma\_write
  - DM7820 driver functions, [17](#)
- dm7820\_enable\_plx\_interrupts
  - DM7820 driver functions, [17](#)
- dm7820\_free\_dma\_mappings
  - DM7820 driver functions, [17](#)
- dm7820\_get\_buffer\_phy\_addr
  - DM7820 driver functions, [18](#)
- dm7820\_get\_interrupt\_status
  - DM7820 driver functions, [18](#)
- dm7820\_get\_pci\_master\_status
  - DM7820 driver functions, [18](#)
- dm7820\_initialize\_device\_descriptor
  - DM7820 driver functions, [19](#)
- dm7820\_initialize\_dma
  - DM7820 driver functions, [19](#)
- dm7820\_initialize\_hardware
  - DM7820 driver functions, [20](#)
- dm7820\_int\_queue\_add
  - DM7820 driver functions, [20](#)
- dm7820\_interrupt\_control, [152](#)
  - int\_enable\_bit, [152](#)
  - int\_status\_bit, [152](#)
  - minor\_enable, [153](#)
  - minor\_reg, [153](#)
  - minor\_status, [153](#)
- dm7820\_interrupt\_handler
  - DM7820 driver functions, [20](#)
- dm7820\_interrupt\_status\_source, [153](#)
  - minor\_reg, [153](#)
  - source, [153](#)
  - source\_table, [154](#)
- dm7820\_ioctl
  - DM7820 driver functions, [21](#)
- dm7820\_ioctl\_argument, [154](#)
  - dma\_function, [154](#)
  - int\_status, [154](#)
  - modify, [154](#)
  - readwrite, [155](#)
- dm7820\_ioctl\_dma\_function, [155](#)
  - arguments, [155](#)
  - buffer\_address, [155](#)
  - DMA\_Transfer\_Size, [156](#)
  - direction, [155](#)
  - fifo, [156](#)
  - function, [156](#)
  - transfer\_size, [156](#)
  - user\_buffer, [156](#)
- dm7820\_ioctl\_interrupt\_status, [156](#)
  - int\_source\_info, [157](#)
  - wait\_for\_interrupt, [157](#)
- dm7820\_ioctl\_region\_modify, [157](#)
  - access, [157](#)
  - mask, [158](#)
  - mask16, [158](#)
  - mask32, [158](#)
  - mask8, [158](#)
- dm7820\_ioctl\_region\_readwrite, [158](#)
  - access, [159](#)
- dm7820\_ioctl\_region\_readwrite\_t
  - DM7820 ioctl structures, [33](#)
- dm7820\_load
  - DM7820 driver functions, [21](#)
- dm7820\_minor\_int\_reg\_layout, [159](#)
  - enable\_mask, [159](#)
  - offset, [159](#)
  - reserved\_mask, [159](#)
  - status\_mask, [159](#)
- dm7820\_modify\_pci\_region
  - DM7820 driver functions, [21](#)
- dm7820\_open
  - DM7820 driver functions, [22](#)
- dm7820\_pci\_access\_request, [160](#)
  - data, [160](#)
  - data16, [160](#)
  - data32, [160](#)
  - data8, [161](#)
  - offset, [161](#)
  - region, [161](#)
  - size, [161](#)
- dm7820\_pci\_access\_request\_t
  - DM7820 type definition structures, [142](#)
- dm7820\_pci\_region, [161](#)
  - allocated, [162](#)
  - io\_addr, [162](#)
  - length, [162](#)
  - phys\_addr, [162](#)
  - virt\_addr, [162](#)
- dm7820\_pci\_region\_access\_dir
  - DM7820 driver enumerations, [8](#)
- dm7820\_pci\_region\_access\_size
  - DM7820 type PCI enumerations, [113](#)
- dm7820\_pci\_region\_num
  - DM7820 type PCI enumerations, [113](#)
- dm7820\_poll
  - DM7820 driver functions, [22](#)
- dm7820\_probe\_device\_blocks
  - DM7820 driver functions, [23](#)
- dm7820\_probe\_devices
  - DM7820 driver functions, [23](#)
- dm7820\_process\_pci\_regions
  - DM7820 driver functions, [24](#)
- dm7820\_read\_pci\_region
  - DM7820 driver functions, [25](#)
- dm7820\_register\_char\_device
  - DM7820 driver functions, [25](#)
- dm7820\_release

- DM7820 driver functions, 26
- dm7820\_release\_resources
  - DM7820 driver functions, 26
- dm7820\_service\_dma\_function
  - DM7820 driver functions, 26
- dm7820\_unregister\_char\_device
  - DM7820 driver functions, 27
- dm7820\_validate\_device
  - DM7820 driver functions, 27
- dm7820\_validate\_pci\_access
  - DM7820 driver functions, 27
- dm7820\_write\_pci\_region
  - DM7820 driver functions, 28
- dma\_buffer
  - dm7820\_dma\_list\_item\_t, 152
- dma\_buffers\_post\_transfer
  - dm7820\_device\_descriptor, 147
- dma\_buffers\_pre\_transfer
  - dm7820\_device\_descriptor, 147
- dma\_capture\_buffers.c
  - BUF\_NUM, 181
  - BUF\_SIZE, 181
  - board, 182
  - DMA\_BUF\_SIZE, 181
  - ISR, 182
  - interrupts, 183
  - main, 182
  - program\_name, 183
  - SAMPLES, 181
- dma\_capture\_file.c
  - BUF\_NUM, 184
  - BUF\_SIZE, 184
  - board, 186
  - DAT\_FILE, 184
  - DMA\_BUF\_SIZE, 185
  - dma\_done\_interrupts, 186
  - ISR, 185
  - main, 185
  - program\_name, 186
  - SAMPLES, 185
  - test\_data, 186
- dma\_done\_interrupts
  - dma\_capture\_file.c, 186
  - loopback.c, 208
  - pwm\_measure.c, 216
- dma\_function
  - dm7820\_ioctl\_argument, 154
- dma\_in\_read\_direction
  - dm7820\_device\_descriptor, 148
- dma\_init
  - dm7820\_dma\_function\_arguments, 150
- dma\_initialized
  - dm7820\_device\_descriptor, 148
- dma\_size
  - dm7820\_device\_descriptor, 148
- dma\_wait\_queue
  - dm7820\_device\_descriptor, 148
- enable\_mask
  - dm7820\_minior\_int\_reg\_layout, 159
- error
  - \_dm7820\_interrupt\_info, 145
- examples/advint\_event\_interrupt.c, 165
- examples/advint\_match\_interrupt.c, 167
- examples/advint\_status.c, 169
- examples/advint\_strobe\_interrupt.c, 171
- examples/basic\_test.c, 173
- examples/brdctl\_device\_info.c, 179
- examples/dma\_capture\_buffers.c, 180
- examples/dma\_capture\_file.c, 183
- examples/fifo\_cascaded.c, 186
- examples/fifo\_interrupt.c, 189
- examples/fifo\_pci\_access.c, 191
- examples/incenc\_encoders.c, 194
- examples/incenc\_interrupt.c, 196
- examples/incenc\_status.c, 198
- examples/library\_test.c, 200
- examples/loopback.c, 205
- examples/prgclk\_clocks.c, 209
- examples/pwm\_interrupt.c, 210
- examples/pwm\_measure.c, 212
- examples/pwm\_modulators.c, 216
- examples/stdio\_digital\_io.c, 218
- examples/stdio\_peripheral\_output.c, 220
- examples/stdio\_strobe\_signal.c, 221
- examples/tmrctr\_interrupt.c, 222
- examples/tmrctr\_status.c, 225
- examples/tmrctr\_timers.c, 228
- exit\_program
  - incenc\_encoders.c, 196
  - pwm\_measure.c, 216
  - tmrctr\_interrupt.c, 225
  - tmrctr\_timers.c, 230
- expect\_failure\_and\_check
  - basic\_test.c, 176
  - library\_test.c, 203
- expect\_success
  - basic\_test.c, 176
  - library\_test.c, 203
- expected
  - register\_read, 163
- fifo
  - dm7820\_ioctl\_dma\_function, 156
- fifo0\_empty\_interrupts
  - loopback.c, 208
- fifo\_cascaded.c
  - get\_fifo\_status, 188
  - main, 188
  - program\_name, 189
- fifo\_interrupt.c
  - main, 190
  - program\_name, 191
- fifo\_pci\_access.c
  - get\_fifo\_status, 193
  - MODULUS\_DIVISOR, 192
  - main, 193
  - program\_name, 193

- file\_descriptor
  - DM7820\_Board\_Descriptor, 146
- function
  - dm7820\_ioctl\_dma\_function, 156
- get\_fifo\_status
  - fifo\_cascaded.c, 188
  - fifo\_pci\_access.c, 193
- INIT\_BAD\_DEVICE\_FILE\_CREATED
  - basic\_test.c, 175
  - library\_test.c, 202
- INIT\_BOARD\_ARRAY\_ALLOCATED
  - basic\_test.c, 175
  - library\_test.c, 202
- INIT\_BOARD\_ARRAY\_OPENED
  - basic\_test.c, 175
  - library\_test.c, 202
- INIT\_NO\_INITIALIZATION
  - basic\_test.c, 175
  - library\_test.c, 202
- ISR
  - advint\_event\_interrupt.c, 166
  - dma\_capture\_buffers.c, 182
  - dma\_capture\_file.c, 185
  - loopback.c, 207
  - pwm\_measure.c, 215
  - tmrctr\_interrupt.c, 224
- incenc\_encoders.c
  - exit\_program, 196
  - main, 195
  - program\_name, 196
  - sigint\_handler, 196
- incenc\_interrupt.c
  - main, 198
  - program\_name, 198
- incenc\_status.c
  - main, 199
  - program\_name, 200
- include/dm7820\_driver.h, 230
- include/dm7820\_globals.h, 234
- include/dm7820\_ioctl.h, 234
- include/dm7820\_library.h, 236
- include/dm7820\_registers.h, 241
- include/dm7820\_types.h, 249
- init\_state
  - basic\_test.c, 177
  - library\_test.c, 205
- initialization\_state
  - basic\_test.c, 175
  - library\_test.c, 202
- initialization\_state\_t
  - basic\_test.c, 175
  - library\_test.c, 202
- int\_enable\_bit
  - dm7820\_interrupt\_control, 152
- int\_missed
  - \_dm7820\_interrupt\_info, 145
- int\_queue\_count
  - dm7820\_device\_descriptor, 148
- int\_queue\_in
  - dm7820\_device\_descriptor, 148
- int\_queue\_missed
  - dm7820\_device\_descriptor, 148
- int\_queue\_out
  - dm7820\_device\_descriptor, 148
- int\_remaining
  - \_dm7820\_interrupt\_info, 145
- int\_source\_info
  - dm7820\_ioctl\_interrupt\_status, 157
- int\_source\_status
  - dm7820\_device\_descriptor, 148
- int\_status
  - dm7820\_device\_descriptor, 149
  - dm7820\_ioctl\_argument, 154
- int\_status\_bit
  - dm7820\_interrupt\_control, 152
- int\_wait\_queue
  - dm7820\_device\_descriptor, 149
- interrupt\_occurred
  - dm7820\_device\_descriptor, 149
- interrupts
  - dma\_capture\_buffers.c, 183
  - tmrctr\_interrupt.c, 225
- io\_addr
  - dm7820\_pci\_region, 162
- irq\_number
  - dm7820\_device\_descriptor, 149
- isr
  - DM7820\_Board\_Descriptor, 146
- length
  - dm7820\_pci\_region, 162
- library\_test.c
  - INIT\_BAD\_DEVICE\_FILE\_CREATED, 202
  - INIT\_BOARD\_ARRAY\_ALLOCATED, 202
  - INIT\_BOARD\_ARRAY\_OPENED, 202
  - INIT\_NO\_INITIALIZATION, 202
- library\_test.c
  - bad\_device\_name, 204
  - boards, 205
  - device\_count, 205
  - expect\_failure\_and\_check, 203
  - expect\_success, 203
  - init\_state, 205
  - initialization\_state, 202
  - initialization\_state\_t, 202
  - main, 203
  - program\_name, 205
- list
  - dm7820\_dma\_list\_item\_t, 152
- loopback.c
  - BUF\_NUM, 207
  - BUF\_SIZE, 207
  - board, 208
  - DMA\_BUF\_SIZE, 207
  - dma\_done\_interrupts, 208
  - fifo0\_empty\_interrupts, 208

- ISR, [207](#)
- main, [207](#)
- program\_name, [208](#)
- SAMPLES, [207](#)
- MODULUS\_DIVISOR
  - fifo\_pci\_access.c, [192](#)
- main
  - advint\_event\_interrupt.c, [166](#)
  - advint\_match\_interrupt.c, [168](#)
  - advint\_status.c, [170](#)
  - advint\_strobe\_interrupt.c, [172](#)
  - basic\_test.c, [176](#)
  - brdctl\_device\_info.c, [179](#)
  - dma\_capture\_buffers.c, [182](#)
  - dma\_capture\_file.c, [185](#)
  - fifo\_cascaded.c, [188](#)
  - fifo\_interrupt.c, [190](#)
  - fifo\_pci\_access.c, [193](#)
  - incenc\_encoders.c, [195](#)
  - incenc\_interrupt.c, [198](#)
  - incenc\_status.c, [199](#)
  - library\_test.c, [203](#)
  - loopback.c, [207](#)
  - prgclk\_clocks.c, [210](#)
  - pwm\_interrupt.c, [212](#)
  - pwm\_measure.c, [215](#)
  - pwm\_modulators.c, [218](#)
  - stdio\_digital\_io.c, [219](#)
  - stdio\_peripheral\_output.c, [220](#)
  - stdio\_strobe\_signal.c, [222](#)
  - tmrctr\_interrupt.c, [224](#)
  - tmrctr\_status.c, [227](#)
  - tmrctr\_timers.c, [229](#)
- mask
  - dm7820\_ioctl\_region\_modify, [158](#)
- mask16
  - dm7820\_ioctl\_region\_modify, [158](#)
- mask32
  - dm7820\_ioctl\_region\_modify, [158](#)
- mask8
  - dm7820\_ioctl\_region\_modify, [158](#)
- minor\_enable
  - dm7820\_interrupt\_control, [153](#)
- minor\_reg
  - dm7820\_interrupt\_control, [153](#)
  - dm7820\_interrupt\_status\_source, [153](#)
- minor\_status
  - dm7820\_interrupt\_control, [153](#)
- modify
  - dm7820\_ioctl\_argument, [154](#)
- offset
  - dm7820\_minor\_int\_reg\_layout, [159](#)
  - dm7820\_pci\_access\_request, [161](#)
  - register\_read, [163](#)
- PLX9056\_DMA\_BURST
  - DM7820 register BAR2 (memory-mapped FPGA registers) offsets, [84](#)
- PLX9056\_DMA\_READY
  - DM7820 register BAR2 (memory-mapped FPGA registers) offsets, [85](#)
- PLX9056\_DMA\_WIDTH\_16
  - DM7820 register BAR2 (memory-mapped FPGA registers) offsets, [86](#)
- PLX9056\_DMA\_WIDTH\_32
  - DM7820 register BAR2 (memory-mapped FPGA registers) offsets, [86](#)
- PLX9056\_DMA\_WIDTH\_8
  - DM7820 register BAR2 (memory-mapped FPGA registers) offsets, [86](#)
- pci
  - dm7820\_device\_descriptor, [149](#)
- phys\_addr
  - dm7820\_pci\_region, [162](#)
- pid
  - DM7820\_Board\_Descriptor, [146](#)
- prgclk\_clocks.c
  - main, [210](#)
  - program\_name, [210](#)
- program\_name
  - advint\_event\_interrupt.c, [167](#)
  - advint\_match\_interrupt.c, [169](#)
  - advint\_status.c, [171](#)
  - advint\_strobe\_interrupt.c, [173](#)
  - basic\_test.c, [177](#)
  - brdctl\_device\_info.c, [180](#)
  - dma\_capture\_buffers.c, [183](#)
  - dma\_capture\_file.c, [186](#)
  - fifo\_cascaded.c, [189](#)
  - fifo\_interrupt.c, [191](#)
  - fifo\_pci\_access.c, [193](#)
  - incenc\_encoders.c, [196](#)
  - incenc\_interrupt.c, [198](#)
  - incenc\_status.c, [200](#)
  - library\_test.c, [205](#)
  - loopback.c, [208](#)
  - prgclk\_clocks.c, [210](#)
  - pwm\_interrupt.c, [212](#)
  - pwm\_measure.c, [216](#)
  - pwm\_modulators.c, [218](#)
  - stdio\_digital\_io.c, [219](#)
  - stdio\_peripheral\_output.c, [221](#)
  - stdio\_strobe\_signal.c, [222](#)
  - tmrctr\_interrupt.c, [225](#)
  - tmrctr\_status.c, [228](#)
  - tmrctr\_timers.c, [230](#)
- pwm\_interrupt.c
  - main, [212](#)
  - program\_name, [212](#)
- pwm\_measure.c
  - BUF\_NUM, [214](#)
  - BUF\_SIZE, [214](#)
  - board, [216](#)
  - DMA\_BUF\_SIZE, [215](#)



- dma\_done\_interrupts, 216
  - exit\_program, 216
  - ISR, 215
  - main, 215
  - program\_name, 216
  - SAMPLES, 215
  - sigint\_handler, 216
  - UTC\_RATE, 215
- pwm\_modulators.c
  - main, 218
  - program\_name, 218
- readwrite
  - dm7820\_ioctl\_argument, 155
- reference\_count
  - dm7820\_device\_descriptor, 149
- region
  - dm7820\_pci\_access\_request, 161
  - register\_read, 163
- region\_names
  - basic\_test.c, 178
- region\_sizes
  - basic\_test.c, 178
- register\_read, 162
  - expected, 163
  - offset, 163
  - region, 163
  - size, 163
- register\_read\_t
  - basic\_test.c, 175
- register\_reads
  - basic\_test.c, 178
- remove\_isr\_flag
  - dm7820\_device\_descriptor, 149
- reserved\_mask
  - dm7820\_minor\_int\_reg\_layout, 159
- SAMPLES
  - dma\_capture\_buffers.c, 181
  - dma\_capture\_file.c, 185
  - loopback.c, 207
  - pwm\_measure.c, 215
- sigint\_handler
  - incenc\_encoders.c, 196
  - pwm\_measure.c, 216
  - tmrctr\_interrupt.c, 224
  - tmrctr\_timers.c, 229
- size
  - dm7820\_pci\_access\_request, 161
  - register\_read, 163
- source
  - \_dm7820\_interrupt\_info, 145
  - dm7820\_interrupt\_status\_source, 153
- source\_table
  - dm7820\_interrupt\_status\_source, 154
- status\_mask
  - dm7820\_minor\_int\_reg\_layout, 159
- stdio\_digital\_io.c
  - main, 219
  - program\_name, 219
- stdio\_peripheral\_output.c
  - main, 220
  - program\_name, 221
- stdio\_strobe\_signal.c
  - main, 222
  - program\_name, 222
- TIMER\_A0\_DIVISOR
  - tmrctr\_status.c, 226
- TIMER\_A1\_DIVISOR
  - tmrctr\_status.c, 226
- TIMER\_A1\_FREQUENCY
  - tmrctr\_status.c, 226
- TIMER\_A2\_DIVISOR
  - tmrctr\_status.c, 227
- test\_data
  - dma\_capture\_file.c, 186
- tmrctr\_interrupt.c
  - exit\_program, 225
  - ISR, 224
  - interrupts, 225
  - main, 224
  - program\_name, 225
  - sigint\_handler, 224
- tmrctr\_status.c
  - disable\_timers, 227
  - main, 227
  - program\_name, 228
  - TIMER\_A0\_DIVISOR, 226
  - TIMER\_A1\_DIVISOR, 226
  - TIMER\_A1\_FREQUENCY, 226
  - TIMER\_A2\_DIVISOR, 227
- tmrctr\_timers.c
  - exit\_program, 230
  - main, 229
  - program\_name, 230
  - sigint\_handler, 229
- transfer\_size
  - dm7820\_ioctl\_dma\_function, 156
- UTC\_RATE
  - pwm\_measure.c, 215
- user\_buffer
  - dm7820\_ioctl\_dma\_function, 156
- virt\_addr
  - dm7820\_pci\_region, 162
- virtual\_address
  - dm7820\_dma\_descriptor\_t, 150
- wait\_for\_interrupt
  - dm7820\_ioctl\_interrupt\_status, 157