

lwIP

1.4.1

Generated by Doxygen 1.8.9.1

Sat Feb 13 2016 16:13:47

Contents

1	Todo List	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	9
4.1	chap_state Struct Reference	9
4.1.1	Detailed Description	9
4.1.2	Field Documentation	9
4.1.2.1	chal_id	9
4.1.2.2	chal_interval	10
4.1.2.3	chal_len	10
4.1.2.4	chal_name	10
4.1.2.5	chal_transmits	10
4.1.2.6	chal_type	10
4.1.2.7	challenge	10
4.1.2.8	clientstate	10
4.1.2.9	id	10
4.1.2.10	max_transmits	10
4.1.2.11	resp_id	10
4.1.2.12	resp_length	10
4.1.2.13	resp_name	10
4.1.2.14	resp_transmits	11
4.1.2.15	resp_type	11
4.1.2.16	response	11
4.1.2.17	serverstate	11
4.1.2.18	timeouttime	11
4.1.2.19	unit	11
4.2	cstate Struct Reference	11

4.2.1	Detailed Description	12
4.2.2	Field Documentation	12
4.2.2.1	cs_filler	12
4.2.2.2	cs_hlen	12
4.2.2.3	cs_id	12
4.2.2.4	cs_next	12
4.2.2.5	csu_hdr	12
4.2.2.6	csu_ip	12
4.2.2.7	vjcs_u	12
4.3	fsm Struct Reference	13
4.3.1	Detailed Description	13
4.3.2	Field Documentation	13
4.3.2.1	callbacks	13
4.3.2.2	flags	13
4.3.2.3	id	14
4.3.2.4	maxconfreqtransmits	14
4.3.2.5	maxnakloops	14
4.3.2.6	maxtermtransmits	14
4.3.2.7	nakloops	14
4.3.2.8	protocol	14
4.3.2.9	reqid	14
4.3.2.10	retransmits	14
4.3.2.11	seen_ack	14
4.3.2.12	state	14
4.3.2.13	term_reason	14
4.3.2.14	term_reason_len	14
4.3.2.15	timeouttime	15
4.3.2.16	unit	15
4.4	fsm_callbacks Struct Reference	15
4.4.1	Detailed Description	15
4.4.2	Field Documentation	15
4.4.2.1	ackci	15
4.4.2.2	addci	15
4.4.2.3	cilen	15
4.4.2.4	down	16
4.4.2.5	extcode	16
4.4.2.6	finished	16
4.4.2.7	nakci	16
4.4.2.8	proto_name	16
4.4.2.9	protreject	16

4.4.2.10	rejci	16
4.4.2.11	reqci	16
4.4.2.12	resetci	16
4.4.2.13	retransmit	16
4.4.2.14	starting	16
4.4.2.15	up	16
4.5	icmp_echo_hdr Struct Reference	17
4.5.1	Detailed Description	17
4.5.2	Member Function Documentation	17
4.5.2.1	PACK_STRUCT_FIELD	17
4.5.2.2	PACK_STRUCT_FIELD	17
4.5.2.3	PACK_STRUCT_FIELD	17
4.5.2.4	PACK_STRUCT_FIELD	17
4.5.2.5	PACK_STRUCT_FIELD	17
4.6	in_addr Struct Reference	17
4.6.1	Detailed Description	17
4.6.2	Field Documentation	18
4.6.2.1	s_addr	18
4.7	ip_addr Struct Reference	18
4.7.1	Detailed Description	18
4.7.2	Member Function Documentation	18
4.7.2.1	PACK_STRUCT_FIELD	18
4.7.3	Field Documentation	18
4.7.3.1	addr	18
4.8	ip_addr2 Struct Reference	18
4.8.1	Detailed Description	19
4.8.2	Member Function Documentation	19
4.8.2.1	PACK_STRUCT_FIELD	19
4.8.2.2	PACK_STRUCT_FIELD	19
4.9	ip_addr_packed Struct Reference	19
4.9.1	Detailed Description	19
4.9.2	Member Function Documentation	19
4.9.2.1	PACK_STRUCT_FIELD	19
4.10	ip_hdr Struct Reference	19
4.10.1	Detailed Description	20
4.10.2	Member Function Documentation	20
4.10.2.1	PACK_STRUCT_FIELD	20
4.10.2.2	PACK_STRUCT_FIELD	20
4.10.2.3	PACK_STRUCT_FIELD	21
4.10.2.4	PACK_STRUCT_FIELD	21

4.10.2.5	PACK_STRUCT_FIELD	21
4.10.2.6	PACK_STRUCT_FIELD	21
4.10.2.7	PACK_STRUCT_FIELD	21
4.10.2.8	PACK_STRUCT_FIELD	21
4.10.2.9	PACK_STRUCT_FIELD	21
4.10.2.10	PACK_STRUCT_FIELD	21
4.10.3	Field Documentation	21
4.10.3.1	dest	21
4.10.3.2	flow1	21
4.10.3.3	flow2	21
4.10.3.4	hoplim	21
4.10.3.5	len	21
4.10.3.6	nexthdr	21
4.10.3.7	tclass1	21
4.10.3.8	tclass2	21
4.10.3.9	v	22
4.11	ip_pcb Struct Reference	22
4.11.1	Detailed Description	22
4.11.2	Field Documentation	22
4.11.2.1	IP_PCB	22
4.12	ipcp_options Struct Reference	22
4.12.1	Detailed Description	23
4.12.2	Field Documentation	23
4.12.2.1	accept_local	23
4.12.2.2	accept_remote	23
4.12.2.3	cflag	23
4.12.2.4	default_route	23
4.12.2.5	dnsaddr	23
4.12.2.6	hisaddr	23
4.12.2.7	maxslotindex	23
4.12.2.8	neg_addr	23
4.12.2.9	neg_vj	23
4.12.2.10	old_addrs	23
4.12.2.11	old_vj	24
4.12.2.12	ouraddr	24
4.12.2.13	proxy_arp	24
4.12.2.14	req_addr	24
4.12.2.15	req_dns1	24
4.12.2.16	req_dns2	24
4.12.2.17	vj_protocol	24

4.12.2.18 winsaddr	24
4.13 lcp_options Struct Reference	24
4.13.1 Detailed Description	25
4.13.2 Field Documentation	25
4.13.2.1 asyncmap	25
4.13.2.2 chap_mdtype	25
4.13.2.3 lqr_period	25
4.13.2.4 magicnumber	25
4.13.2.5 mru	25
4.13.2.6 neg_accompression	25
4.13.2.7 neg_asyncmap	25
4.13.2.8 neg_cbcp	25
4.13.2.9 neg_chap	26
4.13.2.10 neg_lqr	26
4.13.2.11 neg_magicnumber	26
4.13.2.12 neg_mru	26
4.13.2.13 neg_pcompression	26
4.13.2.14 neg_upap	26
4.13.2.15 numloops	26
4.13.2.16 passive	26
4.13.2.17 restart	26
4.13.2.18 silent	26
4.14 MD5_CTX Struct Reference	26
4.14.1 Detailed Description	27
4.14.2 Field Documentation	27
4.14.2.1 buf	27
4.14.2.2 digest	27
4.14.2.3 i	27
4.14.2.4 in	27
4.15 mem Struct Reference	27
4.15.1 Detailed Description	27
4.15.2 Field Documentation	28
4.15.2.1 next	28
4.15.2.2 prev	28
4.15.2.3 used	28
4.16 memp Struct Reference	28
4.16.1 Detailed Description	28
4.16.2 Field Documentation	28
4.16.2.1 next	28
4.17 netbuf Struct Reference	29

4.17.1 Detailed Description	29
4.17.2 Field Documentation	29
4.17.2.1 addr	29
4.17.2.2 p	29
4.17.2.3 port	29
4.17.2.4 ptr	29
4.18 netif Struct Reference	30
4.18.1 Detailed Description	30
4.18.2 Field Documentation	30
4.18.2.1 flags	30
4.18.2.2 gw	31
4.18.2.3 hwaddr	31
4.18.2.4 hwaddr_len	31
4.18.2.5 input	31
4.18.2.6 ip_addr	31
4.18.2.7 linkoutput	31
4.18.2.8 mtu	31
4.18.2.9 name	31
4.18.2.10 netmask	31
4.18.2.11 next	31
4.18.2.12 num	32
4.18.2.13 output	32
4.18.2.14 state	32
4.19 pbuf Struct Reference	32
4.19.1 Detailed Description	32
4.19.2 Field Documentation	33
4.19.2.1 flags	33
4.19.2.2 len	33
4.19.2.3 next	33
4.19.2.4 payload	33
4.19.2.5 ref	33
4.19.2.6 tot_len	33
4.19.2.7 type	33
4.20 sys_timeo Struct Reference	33
4.20.1 Detailed Description	34
4.20.2 Field Documentation	34
4.20.2.1 arg	34
4.20.2.2 h	34
4.20.2.3 next	34
4.20.2.4 time	34

4.21	tcpip_msg Struct Reference	34
4.21.1	Detailed Description	35
4.21.2	Field Documentation	35
4.21.2.1	cb	35
4.21.2.2	ctx	35
4.21.2.3	function	36
4.21.2.4	inp	36
4.21.2.5	msg	36
4.21.2.6	netif	36
4.21.2.7	p	36
4.21.2.8	sem	36
4.21.2.9	type	36
4.22	vjcompress Struct Reference	36
4.22.1	Detailed Description	37
4.22.2	Field Documentation	37
4.22.2.1	compressSlot	37
4.22.2.2	flags	37
4.22.2.3	last_cs	38
4.22.2.4	last_rcv	38
4.22.2.5	last_xmit	38
4.22.2.6	maxSlotIndex	38
4.22.2.7	rstate	38
4.22.2.8	tstate	38
4.23	vjstat Struct Reference	38
4.23.1	Detailed Description	38
4.23.2	Field Documentation	39
4.23.2.1	vjs_compressed	39
4.23.2.2	vjs_compressedin	39
4.23.2.3	vjs_errorin	39
4.23.2.4	vjs_misses	39
4.23.2.5	vjs_packets	39
4.23.2.6	vjs_searches	39
4.23.2.7	vjs_tossed	39
4.23.2.8	vjs_uncompressedin	39
5	File Documentation	41
5.1	src/api/api_lib.c File Reference	41
5.1.1	Detailed Description	41
5.2	src/api/api_msg.c File Reference	42
5.2.1	Detailed Description	42

5.3	src/api/err.c File Reference	43
5.3.1	Detailed Description	43
5.4	src/api/netbuf.c File Reference	44
5.4.1	Detailed Description	44
5.5	src/api/netdb.c File Reference	45
5.5.1	Detailed Description	45
5.6	src/api/netifapi.c File Reference	46
5.6.1	Detailed Description	46
5.7	src/api/sockets.c File Reference	47
5.7.1	Detailed Description	47
5.8	src/api/tcpip.c File Reference	47
5.8.1	Detailed Description	48
5.8.2	Function Documentation	48
5.8.2.1	mem_free_callback	48
5.8.2.2	pbuf_free_callback	48
5.8.2.3	tcpip_callback_with_block	49
5.8.2.4	tcpip_callbackmsg_delete	49
5.8.2.5	tcpip_callbackmsg_new	49
5.8.2.6	tcpip_init	49
5.8.2.7	tcpip_input	50
5.8.2.8	tcpip_trycallback	50
5.9	src/core/def.c File Reference	50
5.9.1	Detailed Description	51
5.9.2	Function Documentation	51
5.9.2.1	lwip_htonl	51
5.9.2.2	lwip_htons	52
5.9.2.3	lwip_ntohl	52
5.9.2.4	lwip_ntohs	52
5.10	src/core/dhcp.c File Reference	53
5.10.1	Detailed Description	53
5.11	src/core/dns.c File Reference	54
5.11.1	Detailed Description	54
5.12	src/core/init.c File Reference	55
5.12.1	Detailed Description	55
5.12.2	Macro Definition Documentation	55
5.12.2.1	LWIP_DISABLE_MEMP_SANITY_CHECKS	55
5.12.2.2	LWIP_DISABLE_TCP_SANITY_CHECKS	56
5.12.3	Function Documentation	56
5.12.3.1	lwip_init	56
5.13	src/core/ipv4/autoip.c File Reference	56

5.13.1 Detailed Description	56
5.14 src/core/ipv4/icmp.c File Reference	57
5.14.1 Detailed Description	57
5.15 src/core/ipv4/igmp.c File Reference	58
5.15.1 Detailed Description	58
5.16 src/core/ipv4/inet.c File Reference	58
5.16.1 Detailed Description	59
5.17 src/core/ipv4/inet_chksum.c File Reference	59
5.17.1 Detailed Description	60
5.17.2 Macro Definition Documentation	60
5.17.2.1 LWIP_CHKSUM	61
5.17.2.2 LWIP_CHKSUM_ALGORITHM	61
5.17.3 Function Documentation	61
5.17.3.1 inet_chksum	61
5.17.3.2 inet_chksum_pbuf	61
5.17.3.3 inet_chksum_pseudo	61
5.17.3.4 inet_chksum_pseudo_partial	61
5.18 src/core/ipv4/ip.c File Reference	62
5.18.1 Detailed Description	63
5.18.2 Macro Definition Documentation	63
5.18.2.1 CHECKSUM_GEN_IP_INLINE	63
5.18.2.2 IP_ACCEPT_LINK_LAYER_ADDRESSING	63
5.18.2.3 LWIP_INLINE_IP_CHKSUM	63
5.18.3 Function Documentation	63
5.18.3.1 ip_input	63
5.18.3.2 ip_output	63
5.18.3.3 ip_output_if	64
5.18.3.4 ip_route	64
5.18.4 Variable Documentation	65
5.18.4.1 current_header	65
5.18.4.2 current_iphdr_dest	65
5.18.4.3 current_iphdr_src	65
5.18.4.4 current_netif	65
5.19 src/core/ipv4/ip_addr.c File Reference	65
5.19.1 Detailed Description	67
5.19.2 Macro Definition Documentation	67
5.19.2.1 in_range	67
5.19.2.2 isdigit	67
5.19.2.3 islower	67
5.19.2.4 isprint	67

5.19.2.5	isspace	67
5.19.2.6	isxdigit	67
5.19.3	Function Documentation	67
5.19.3.1	ip4_addr_isbroadcast	67
5.19.3.2	ip4_addr_netmask_valid	68
5.19.3.3	ipaddr_addr	69
5.19.3.4	ipaddr_aton	69
5.19.3.5	ipaddr_ntoa	69
5.19.3.6	ipaddr_ntoa_r	69
5.19.4	Variable Documentation	70
5.19.4.1	ip_addr_any	70
5.19.4.2	ip_addr_broadcast	70
5.20	src/core/ipv4/ip_frag.c File Reference	70
5.20.1	Detailed Description	71
5.21	src/core/ipv6/icmp6.c File Reference	71
5.22	src/core/ipv6/inet6.c File Reference	71
5.22.1	Detailed Description	72
5.22.2	Function Documentation	72
5.22.2.1	inet_chksum	72
5.22.2.2	inet_chksum_pbuf	72
5.22.2.3	inet_chksum_pseudo	73
5.23	src/core/ipv6/ip6.c File Reference	73
5.23.1	Function Documentation	74
5.23.1.1	ip_init	74
5.23.1.2	ip_input	74
5.23.1.3	ip_output	74
5.23.1.4	ip_output_if	74
5.23.1.5	ip_route	74
5.24	src/core/ipv6/ip6_addr.c File Reference	74
5.24.1	Function Documentation	75
5.24.1.1	ip_addr_cmp	75
5.24.1.2	ip_addr_isany	75
5.24.1.3	ip_addr_netcmp	75
5.24.1.4	ip_addr_set	75
5.25	src/core/mem.c File Reference	76
5.25.1	Detailed Description	77
5.25.2	Macro Definition Documentation	77
5.25.2.1	LWIP_MEM_ALLOC_DECL_PROTECT	77
5.25.2.2	LWIP_MEM_ALLOC_PROTECT	77
5.25.2.3	LWIP_MEM_ALLOC_UNPROTECT	77

5.25.2.4	LWIP_MEM_FREE_DECL_PROTECT	77
5.25.2.5	LWIP_MEM_FREE_PROTECT	77
5.25.2.6	LWIP_MEM_FREE_UNPROTECT	77
5.25.2.7	LWIP_RAM_HEAP_POINTER	78
5.25.2.8	MEM_SIZE_ALIGNED	78
5.25.2.9	MIN_SIZE	78
5.25.2.10	MIN_SIZE_ALIGNED	78
5.25.2.11	SIZEOF_STRUCT_MEM	78
5.25.3	Function Documentation	78
5.25.3.1	mem_calloc	78
5.25.3.2	mem_free	78
5.25.3.3	mem_init	79
5.25.3.4	mem_malloc	79
5.25.3.5	mem_trim	79
5.25.4	Variable Documentation	79
5.25.4.1	ram_heap	79
5.26	src/core/memp.c File Reference	79
5.26.1	Detailed Description	80
5.26.2	Macro Definition Documentation	81
5.26.2.1	LWIP_MEMPOOL	81
5.26.2.2	LWIP_MEMPOOL	81
5.26.2.3	LWIP_MEMPOOL	81
5.26.2.4	MEMP_ALIGN_SIZE	81
5.26.2.5	MEMP_SIZE	81
5.26.3	Function Documentation	81
5.26.3.1	memp_free	81
5.26.3.2	memp_init	81
5.26.3.3	memp_malloc	81
5.27	src/core/netif.c File Reference	82
5.27.1	Detailed Description	83
5.27.2	Macro Definition Documentation	83
5.27.2.1	NETIF_LINK_CALLBACK	83
5.27.2.2	NETIF_STATUS_CALLBACK	83
5.27.3	Function Documentation	83
5.27.3.1	netif_add	83
5.27.3.2	netif_find	83
5.27.3.3	netif_init	84
5.27.3.4	netif_remove	84
5.27.3.5	netif_set_addr	84
5.27.3.6	netif_set_default	84

5.27.3.7	netif_set_down	84
5.27.3.8	netif_set_gw	85
5.27.3.9	netif_set_ipaddr	86
5.27.3.10	netif_set_link_down	86
5.27.3.11	netif_set_link_up	86
5.27.3.12	netif_set_netmask	86
5.27.3.13	netif_set_up	86
5.27.4	Variable Documentation	87
5.27.4.1	netif_default	87
5.27.4.2	netif_list	87
5.28	src/core/pbuf.c File Reference	87
5.28.1	Detailed Description	88
5.28.2	Macro Definition Documentation	88
5.28.2.1	PBUF_POOL_BUFSIZE_ALIGNED	88
5.28.2.2	PBUF_POOL_IS_EMPTY	88
5.28.2.3	SIZEOF_STRUCT_PBUF	89
5.28.3	Function Documentation	89
5.28.3.1	pbuf_alloc	89
5.28.3.2	pbuf_cat	89
5.28.3.3	pbuf_chain	89
5.28.3.4	pbuf_clen	90
5.28.3.5	pbuf_coalesce	90
5.28.3.6	pbuf_copy	90
5.28.3.7	pbuf_copy_partial	91
5.28.3.8	pbuf_dechain	91
5.28.3.9	pbuf_free	91
5.28.3.10	pbuf_get_at	92
5.28.3.11	pbuf_header	92
5.28.3.12	pbuf_memcmp	92
5.28.3.13	pbuf_memfind	93
5.28.3.14	pbuf_realloc	93
5.28.3.15	pbuf_ref	93
5.28.3.16	pbuf_strstr	94
5.28.3.17	pbuf_take	94
5.29	src/core/raw.c File Reference	94
5.29.1	Detailed Description	95
5.30	src/core/snmp/asn1_dec.c File Reference	96
5.30.1	Detailed Description	96
5.31	src/core/snmp/asn1_enc.c File Reference	96
5.31.1	Detailed Description	97

5.32	src/core/snmp/mib2.c File Reference	97
5.32.1	Detailed Description	98
5.33	src/core/snmp/mib_structs.c File Reference	98
5.33.1	Detailed Description	99
5.34	src/core/snmp/msg_in.c File Reference	100
5.34.1	Detailed Description	100
5.35	src/core/snmp/msg_out.c File Reference	101
5.35.1	Detailed Description	101
5.36	src/core/stats.c File Reference	101
5.36.1	Detailed Description	102
5.37	src/core/sys.c File Reference	102
5.37.1	Detailed Description	103
5.37.2	Function Documentation	103
5.37.2.1	sys_msleep	103
5.38	src/core/tcp.c File Reference	104
5.38.1	Detailed Description	104
5.39	src/core/tcp_in.c File Reference	104
5.39.1	Detailed Description	105
5.40	src/core/tcp_out.c File Reference	105
5.40.1	Detailed Description	106
5.41	src/core/timers.c File Reference	106
5.41.1	Detailed Description	107
5.41.2	Function Documentation	107
5.41.2.1	tcp_timer_needed	107
5.42	src/core/udp.c File Reference	108
5.42.1	Detailed Description	108
5.43	src/include/ipv4/lwip/autoip.h File Reference	109
5.43.1	Detailed Description	109
5.44	src/include/ipv4/lwip/icmp.h File Reference	110
5.44.1	Macro Definition Documentation	111
5.44.1.1	ICMP_DUR	111
5.44.1.2	ICMP_ECHO	111
5.44.1.3	ICMP_ER	111
5.44.1.4	ICMP_IR	111
5.44.1.5	ICMP_IRQ	111
5.44.1.6	ICMP_PP	111
5.44.1.7	ICMP_RD	112
5.44.1.8	ICMP_SQ	112
5.44.1.9	ICMP_TE	112
5.44.1.10	ICMP_TS	112

5.44.1.11	ICMP_TSR	112
5.44.1.12	ICMPH_CODE	112
5.44.1.13	ICMPH_CODE_SET	112
5.44.1.14	ICMPH_TYPE	112
5.44.1.15	ICMPH_TYPE_SET	112
5.44.2	Enumeration Type Documentation	112
5.44.2.1	icmp_dur_type	112
5.44.2.2	icmp_te_type	113
5.44.3	Variable Documentation	113
5.44.3.1	PACK_STRUCT_STRUCT	113
5.45	src/include/ipv6/lwip/icmp.h File Reference	113
5.46	src/include/ipv4/lwip/igmp.h File Reference	114
5.47	src/include/ipv4/lwip/inet.h File Reference	116
5.47.1	Macro Definition Documentation	117
5.47.1.1	IN_BADCLASS	117
5.47.1.2	IN_CLASSA	117
5.47.1.3	IN_CLASSA_HOST	117
5.47.1.4	IN_CLASSA_MAX	117
5.47.1.5	IN_CLASSA_NET	117
5.47.1.6	IN_CLASSA_NSHIFT	118
5.47.1.7	IN_CLASSB	118
5.47.1.8	IN_CLASSB_HOST	118
5.47.1.9	IN_CLASSB_MAX	118
5.47.1.10	IN_CLASSB_NET	118
5.47.1.11	IN_CLASSB_NSHIFT	118
5.47.1.12	IN_CLASSC	118
5.47.1.13	IN_CLASSC_HOST	118
5.47.1.14	IN_CLASSC_MAX	118
5.47.1.15	IN_CLASSC_NET	118
5.47.1.16	IN_CLASSC_NSHIFT	118
5.47.1.17	IN_CLASSD	118
5.47.1.18	IN_CLASSD_HOST	119
5.47.1.19	IN_CLASSD_MAX	119
5.47.1.20	IN_CLASSD_NET	119
5.47.1.21	IN_CLASSD_NSHIFT	119
5.47.1.22	IN_EXPERIMENTAL	119
5.47.1.23	IN_LOOPBACKNET	119
5.47.1.24	IN_MULTICAST	119
5.47.1.25	INADDR_ANY	119
5.47.1.26	INADDR_BROADCAST	119

5.47.1.27 INADDR_LOOPBACK	119
5.47.1.28 INADDR_NONE	119
5.47.1.29 inet_addr	120
5.47.1.30 inet_addr_from_ipaddr	120
5.47.1.31 inet_addr_to_ipaddr	120
5.47.1.32 inet_addr_to_ipaddr_p	120
5.47.1.33 inet_aton	120
5.47.1.34 inet_ntoa	120
5.47.1.35 inet_ntoa_r	120
5.48 src/include/ipv6/lwip/inet.h File Reference	120
5.48.1 Function Documentation	122
5.48.1.1 htonl	122
5.48.1.2 htons	122
5.48.1.3 inet_addr	122
5.48.1.4 inet_aton	122
5.48.1.5 inet_chksum	122
5.48.1.6 inet_chksum_pbuf	122
5.48.1.7 inet_chksum_pseudo	122
5.48.1.8 ntohl	122
5.48.1.9 ntohs	122
5.49 src/include/ipv4/lwip/inet_chksum.h File Reference	122
5.49.1 Macro Definition Documentation	124
5.49.1.1 FOLD_U32T	124
5.49.1.2 LWIP_CHKSUM_COPY_ALGORITHM	124
5.49.1.3 SWAP_BYTES_IN_WORD	124
5.49.2 Function Documentation	124
5.49.2.1 inet_chksum	124
5.49.2.2 inet_chksum_pbuf	124
5.49.2.3 inet_chksum_pseudo	124
5.49.2.4 inet_chksum_pseudo_partial	125
5.50 src/include/ipv4/lwip/ip.h File Reference	125
5.50.1 Macro Definition Documentation	127
5.50.1.1 ip_current_dest_addr	127
5.50.1.2 ip_current_header	127
5.50.1.3 ip_current_netif	127
5.50.1.4 ip_current_src_addr	127
5.50.1.5 ip_debug_print	127
5.50.1.6 IP_DF	127
5.50.1.7 ip_get_option	128
5.50.1.8 IP_HDRINCL	128

5.50.1.9	IP_HLEN	128
5.50.1.10	ip_init	128
5.50.1.11	IP_MF	128
5.50.1.12	IP_OFFMASK	128
5.50.1.13	IP_OPTIONS_SEND	128
5.50.1.14	IP_PCB	128
5.50.1.15	IP_PCB_ADDRHINT	128
5.50.1.16	IP_PROTO_ICMP	129
5.50.1.17	IP_PROTO_IGMP	129
5.50.1.18	IP_PROTO_TCP	129
5.50.1.19	IP_PROTO_UDP	129
5.50.1.20	IP_PROTO_UDPLITE	129
5.50.1.21	ip_reset_option	129
5.50.1.22	IP_RF	129
5.50.1.23	ip_set_option	129
5.50.1.24	IPH_CHKSUM	129
5.50.1.25	IPH_CHKSUM_SET	129
5.50.1.26	IPH_HL	129
5.50.1.27	IPH_ID	129
5.50.1.28	IPH_ID_SET	130
5.50.1.29	IPH_LEN	130
5.50.1.30	IPH_LEN_SET	130
5.50.1.31	IPH_OFFSET	130
5.50.1.32	IPH_OFFSET_SET	130
5.50.1.33	IPH_PROTO	130
5.50.1.34	IPH_PROTO_SET	130
5.50.1.35	IPH_TOS	130
5.50.1.36	IPH_TOS_SET	130
5.50.1.37	IPH_TTL	130
5.50.1.38	IPH_TTL_SET	130
5.50.1.39	IPH_V	130
5.50.1.40	IPH_VHL_SET	131
5.50.1.41	SOF_ACCEPTCONN	131
5.50.1.42	SOF_BROADCAST	131
5.50.1.43	SOF_INHERITED	131
5.50.1.44	SOF_KEEPAIVE	131
5.50.1.45	SOF_LINGER	131
5.50.1.46	SOF_REUSEADDR	131
5.50.2	Function Documentation	131
5.50.2.1	ip_input	131

5.50.2.2	ip_output	132
5.50.2.3	ip_output_if	133
5.50.2.4	ip_route	133
5.50.3	Variable Documentation	134
5.50.3.1	current_header	134
5.50.3.2	current_iphdr_dest	134
5.50.3.3	current_iphdr_src	134
5.50.3.4	current_netif	134
5.50.3.5	PACK_STRUCT_STRUCT	134
5.51	src/include/ipv6/lwip/ip.h File Reference	134
5.51.1	Macro Definition Documentation	136
5.51.1.1	ip_current_header	136
5.51.1.2	ip_current_netif	136
5.51.1.3	IP_HDRINCL	136
5.51.1.4	IP_HLEN	136
5.51.1.5	IP_PCB	136
5.51.1.6	IP_PCB_ADDRHINT	137
5.51.1.7	IP_PROTO_ICMP	137
5.51.1.8	IP_PROTO_TCP	137
5.51.1.9	IP_PROTO_UDP	137
5.51.1.10	IP_PROTO_UDPLITE	137
5.51.1.11	IPH_PROTO	137
5.51.2	Function Documentation	137
5.51.2.1	ip_init	137
5.51.2.2	ip_input	137
5.51.2.3	ip_output	137
5.51.2.4	ip_output_if	138
5.51.2.5	ip_route	138
5.52	src/include/ipv4/lwip/ip_addr.h File Reference	138
5.52.1	Macro Definition Documentation	140
5.52.1.1	IP4_ADDR	140
5.52.1.2	ip4_addr1	140
5.52.1.3	ip4_addr1_16	140
5.52.1.4	ip4_addr2	140
5.52.1.5	ip4_addr2_16	141
5.52.1.6	ip4_addr3	141
5.52.1.7	ip4_addr3_16	141
5.52.1.8	ip4_addr4	141
5.52.1.9	ip4_addr4_16	141
5.52.1.10	ip4_addr_get_u32	141

5.52.1.11 ip4_addr_set_u32	141
5.52.1.12 IP_ADDR_ANY	141
5.52.1.13 IP_ADDR_BROADCAST	141
5.52.1.14 ip_addr_cmp	141
5.52.1.15 ip_addr_copy	141
5.52.1.16 ip_addr_debug_print	142
5.52.1.17 ip_addr_get_network	142
5.52.1.18 ip_addr_isany	142
5.52.1.19 ip_addr_isbroadcast	142
5.52.1.20 ip_addr_islinklocal	142
5.52.1.21 ip_addr_ismulticast	142
5.52.1.22 ip_addr_netcmp	142
5.52.1.23 ip_addr_netmask_valid	143
5.52.1.24 ip_addr_set	143
5.52.1.25 ip_addr_set_any	143
5.52.1.26 ip_addr_set_hton	143
5.52.1.27 ip_addr_set_loopback	143
5.52.1.28 ip_addr_set_zero	143
5.52.1.29 IP_BADCLASS	143
5.52.1.30 IP_CLASSA	143
5.52.1.31 IP_CLASSA_HOST	144
5.52.1.32 IP_CLASSA_MAX	144
5.52.1.33 IP_CLASSA_NET	144
5.52.1.34 IP_CLASSA_NSHIFT	144
5.52.1.35 IP_CLASSB	144
5.52.1.36 IP_CLASSB_HOST	144
5.52.1.37 IP_CLASSB_MAX	144
5.52.1.38 IP_CLASSB_NET	144
5.52.1.39 IP_CLASSB_NSHIFT	144
5.52.1.40 IP_CLASSC	144
5.52.1.41 IP_CLASSC_HOST	144
5.52.1.42 IP_CLASSC_NET	144
5.52.1.43 IP_CLASSC_NSHIFT	145
5.52.1.44 IP_CLASSD	145
5.52.1.45 IP_CLASSD_HOST	145
5.52.1.46 IP_CLASSD_NET	145
5.52.1.47 IP_CLASSD_NSHIFT	145
5.52.1.48 IP_EXPERIMENTAL	145
5.52.1.49 IP_LOOPBACKNET	145
5.52.1.50 IP_MULTICAST	145

5.52.1.51	ip_ntoa	145
5.52.1.52	IPADDR2_COPY	145
5.52.1.53	IPADDR_ANY	145
5.52.1.54	IPADDR_BROADCAST	146
5.52.1.55	IPADDR_LOOPBACK	146
5.52.1.56	IPADDR_NONE	146
5.52.2	Typedef Documentation	146
5.52.2.1	ip_addr_p_t	146
5.52.2.2	ip_addr_t	146
5.52.3	Function Documentation	146
5.52.3.1	ip4_addr_isbroadcast	146
5.52.3.2	ip4_addr_netmask_valid	146
5.52.3.3	ipaddr_addr	147
5.52.3.4	ipaddr_aton	147
5.52.3.5	ipaddr_ntoa	147
5.52.3.6	ipaddr_ntoa_r	147
5.52.4	Variable Documentation	148
5.52.4.1	ip_addr_any	148
5.52.4.2	ip_addr_broadcast	148
5.52.4.3	PACK_STRUCT_STRUCT	148
5.53	src/include/ipv6/lwip/ip_addr.h File Reference	149
5.53.1	Macro Definition Documentation	150
5.53.1.1	IP6_ADDR	150
5.53.1.2	IP_ADDR_ANY	150
5.53.1.3	ip_addr_debug_print	150
5.53.2	Function Documentation	150
5.53.2.1	ip_addr_cmp	150
5.53.2.2	ip_addr_isany	150
5.53.2.3	ip_addr_netcmp	151
5.53.2.4	ip_addr_set	151
5.53.3	Variable Documentation	151
5.53.3.1	PACK_STRUCT_STRUCT	151
5.54	src/include/ipv4/lwip/ip_frag.h File Reference	151
5.55	src/include/lwip/api.h File Reference	153
5.56	src/include/lwip/api_msg.h File Reference	154
5.57	src/include/lwip/arch.h File Reference	155
5.57.1	Macro Definition Documentation	155
5.57.1.1	BIG_ENDIAN	155
5.57.1.2	LITTLE_ENDIAN	155
5.57.1.3	LWIP_UNUSED_ARG	155

5.57.1.4	PACK_STRUCT_BEGIN	156
5.57.1.5	PACK_STRUCT_END	156
5.57.1.6	PACK_STRUCT_FIELD	156
5.57.1.7	SZT_F	156
5.57.1.8	X8_F	156
5.58	src/include/lwip/debug.h File Reference	156
5.58.1	Macro Definition Documentation	157
5.58.1.1	LWIP_ASSERT	157
5.58.1.2	LWIP_DBG_FRESH	157
5.58.1.3	LWIP_DBG_HALT	157
5.58.1.4	LWIP_DBG_LEVEL_ALL	157
5.58.1.5	LWIP_DBG_LEVEL_OFF	158
5.58.1.6	LWIP_DBG_LEVEL_SERIOUS	158
5.58.1.7	LWIP_DBG_LEVEL_SEVERE	158
5.58.1.8	LWIP_DBG_LEVEL_WARNING	158
5.58.1.9	LWIP_DBG_MASK_LEVEL	158
5.58.1.10	LWIP_DBG_OFF	158
5.58.1.11	LWIP_DBG_ON	158
5.58.1.12	LWIP_DBG_STATE	158
5.58.1.13	LWIP_DBG_TRACE	158
5.58.1.14	LWIP_DEBUGF	158
5.58.1.15	LWIP_ERROR	158
5.59	src/include/lwip/def.h File Reference	159
5.59.1	Macro Definition Documentation	160
5.59.1.1	htonl	160
5.59.1.2	htons	160
5.59.1.3	lwip_htonl	160
5.59.1.4	lwip_htons	160
5.59.1.5	LWIP_MAKE_U16	160
5.59.1.6	LWIP_MAX	160
5.59.1.7	LWIP_MIN	160
5.59.1.8	lwip_ntohl	160
5.59.1.9	lwip_ntohs	161
5.59.1.10	LWIP_PLATFORM_BYTESWAP	161
5.59.1.11	ntohl	161
5.59.1.12	ntohs	161
5.59.1.13	NULL	161
5.59.1.14	PP_HTONL	161
5.59.1.15	PP_HTONS	161
5.59.1.16	PP_NTOHL	161

5.59.1.17 PP_NTOHS	161
5.60 src/include/lwip/dhcp.h File Reference	162
5.61 src/include/lwip/dns.h File Reference	162
5.62 src/include/lwip/err.h File Reference	163
5.62.1 Macro Definition Documentation	165
5.62.1.1 ERR_ABRT	165
5.62.1.2 ERR_ARG	165
5.62.1.3 ERR_BUF	165
5.62.1.4 ERR_CLSD	165
5.62.1.5 ERR_CONN	165
5.62.1.6 ERR_IF	165
5.62.1.7 ERR_INPROGRESS	165
5.62.1.8 ERR_IS_FATAL	165
5.62.1.9 ERR_ISCONN	165
5.62.1.10 ERR_MEM	165
5.62.1.11 ERR_OK	166
5.62.1.12 ERR_RST	166
5.62.1.13 ERR_RTE	166
5.62.1.14 ERR_TIMEOUT	166
5.62.1.15 ERR_USE	166
5.62.1.16 ERR_VAL	166
5.62.1.17 ERR_WOULDBLOCK	166
5.62.1.18 lwip_strerror	166
5.62.2 Typedef Documentation	166
5.62.2.1 err_t	166
5.63 src/include/lwip/init.h File Reference	166
5.63.1 Macro Definition Documentation	168
5.63.1.1 LWIP_RC_DEVELOPMENT	168
5.63.1.2 LWIP_RC_RELEASE	168
5.63.1.3 LWIP_VERSION	168
5.63.1.4 LWIP_VERSION_IS_DEVELOPMENT	168
5.63.1.5 LWIP_VERSION_IS_RC	168
5.63.1.6 LWIP_VERSION_IS_RELEASE	168
5.63.1.7 LWIP_VERSION_MAJOR	168
5.63.1.8 LWIP_VERSION_MINOR	169
5.63.1.9 LWIP_VERSION_RC	169
5.63.1.10 LWIP_VERSION_REVISION	169
5.63.2 Function Documentation	169
5.63.2.1 lwip_init	169
5.64 src/include/lwip/mem.h File Reference	169

5.64.1	Macro Definition Documentation	171
5.64.1.1	LWIP_MEM_ALIGN	171
5.64.1.2	LWIP_MEM_ALIGN_BUFFER	171
5.64.1.3	LWIP_MEM_ALIGN_SIZE	171
5.64.1.4	MEM_SIZE_F	171
5.64.2	Typedef Documentation	171
5.64.2.1	mem_size_t	171
5.64.3	Function Documentation	171
5.64.3.1	mem_calloc	171
5.64.3.2	mem_free	172
5.64.3.3	mem_init	172
5.64.3.4	mem_malloc	172
5.64.3.5	mem_trim	172
5.65	src/include/lwip/memp.h File Reference	173
5.65.1	Macro Definition Documentation	174
5.65.1.1	LWIP_MEMPOOL	174
5.65.2	Enumeration Type Documentation	174
5.65.2.1	memp_t	174
5.65.3	Function Documentation	174
5.65.3.1	memp_free	174
5.65.3.2	memp_init	174
5.65.3.3	memp_malloc	174
5.66	src/include/lwip/memp_std.h File Reference	175
5.66.1	Macro Definition Documentation	175
5.66.1.1	LWIP_MALLOC_MEMPOOL	175
5.66.1.2	LWIP_MALLOC_MEMPOOL_END	175
5.66.1.3	LWIP_MALLOC_MEMPOOL_START	175
5.66.1.4	LWIP_PBUF_MEMPOOL	175
5.67	src/include/lwip/netbuf.h File Reference	176
5.67.1	Macro Definition Documentation	177
5.67.1.1	netbuf_copy	177
5.67.1.2	netbuf_copy_partial	177
5.67.1.3	NETBUF_FLAG_CHKSUM	177
5.67.1.4	NETBUF_FLAG_DESTADDR	177
5.67.1.5	netbuf_fromaddr	177
5.67.1.6	netbuf_fromport	177
5.67.1.7	netbuf_len	177
5.67.1.8	netbuf_set_fromaddr	178
5.67.1.9	netbuf_take	178
5.67.2	Function Documentation	178

5.67.2.1	netbuf_alloc	178
5.67.2.2	netbuf_chain	178
5.67.2.3	netbuf_data	178
5.67.2.4	netbuf_delete	178
5.67.2.5	netbuf_first	178
5.67.2.6	netbuf_free	178
5.67.2.7	netbuf_new	178
5.67.2.8	netbuf_next	178
5.67.2.9	netbuf_ref	178
5.68	src/include/lwip/netdb.h File Reference	179
5.69	src/include/posix/netdb.h File Reference	179
5.69.1	Detailed Description	180
5.70	src/include/lwip/netif.h File Reference	180
5.70.1	Macro Definition Documentation	182
5.70.1.1	ENABLE_LOOPBACK	182
5.70.1.2	NETIF_FLAG_BROADCAST	183
5.70.1.3	NETIF_FLAG_DHCP	183
5.70.1.4	NETIF_FLAG_ETHARP	183
5.70.1.5	NETIF_FLAG_ETHERNET	183
5.70.1.6	NETIF_FLAG_IGMP	183
5.70.1.7	NETIF_FLAG_LINK_UP	183
5.70.1.8	NETIF_FLAG_POINTTOPOINT	183
5.70.1.9	NETIF_FLAG_UP	183
5.70.1.10	NETIF_INIT_SNMP	183
5.70.1.11	netif_is_link_up	184
5.70.1.12	netif_is_up	184
5.70.1.13	NETIF_MAX_HWADDR_LEN	184
5.70.1.14	NETIF_SET_HWADDRHINT	184
5.70.2	Typedef Documentation	184
5.70.2.1	netif_igmp_mac_filter_fn	184
5.70.2.2	netif_init_fn	184
5.70.2.3	netif_input_fn	184
5.70.2.4	netif_linkoutput_fn	184
5.70.2.5	netif_output_fn	185
5.70.2.6	netif_status_callback_fn	185
5.70.3	Function Documentation	185
5.70.3.1	netif_add	185
5.70.3.2	netif_find	185
5.70.3.3	netif_init	186
5.70.3.4	netif_remove	186

5.70.3.5	netif_set_addr	186
5.70.3.6	netif_set_default	186
5.70.3.7	netif_set_down	186
5.70.3.8	netif_set_gw	187
5.70.3.9	netif_set_ipaddr	188
5.70.3.10	netif_set_link_down	188
5.70.3.11	netif_set_link_up	188
5.70.3.12	netif_set_netmask	188
5.70.3.13	netif_set_up	188
5.70.4	Variable Documentation	189
5.70.4.1	netif_default	189
5.70.4.2	netif_list	189
5.71	src/include/lwip/netifapi.h File Reference	189
5.72	src/include/lwip/opt.h File Reference	190
5.72.1	Detailed Description	195
5.72.2	Macro Definition Documentation	195
5.72.2.1	API_LIB_DEBUG	195
5.72.2.2	API_MSG_DEBUG	195
5.72.2.3	ARP_QUEUEING	195
5.72.2.4	ARP_TABLE_SIZE	195
5.72.2.5	AUTOIP_DEBUG	195
5.72.2.6	CHECKSUM_CHECK_IP	195
5.72.2.7	CHECKSUM_CHECK_TCP	195
5.72.2.8	CHECKSUM_CHECK_UDP	195
5.72.2.9	CHECKSUM_GEN_ICMP	196
5.72.2.10	CHECKSUM_GEN_IP	196
5.72.2.11	CHECKSUM_GEN_TCP	196
5.72.2.12	CHECKSUM_GEN_UDP	196
5.72.2.13	DEFAULT_ACCEPTMBOX_SIZE	196
5.72.2.14	DEFAULT_RAW_RECVMBOX_SIZE	196
5.72.2.15	DEFAULT_TCP_RECVMBOX_SIZE	196
5.72.2.16	DEFAULT_THREAD_NAME	196
5.72.2.17	DEFAULT_THREAD_PRIO	196
5.72.2.18	DEFAULT_THREAD_STACKSIZE	197
5.72.2.19	DEFAULT_UDP_RECVMBOX_SIZE	197
5.72.2.20	DHCP_DEBUG	197
5.72.2.21	DHCP_DOES_ARP_CHECK	197
5.72.2.22	DNS_DEBUG	197
5.72.2.23	DNS_DOES_NAME_CHECK	197
5.72.2.24	DNS_LOCAL_HOSTLIST	197

5.72.2.25 DNS_LOCAL_HOSTLIST_IS_DYNAMIC	197
5.72.2.26 DNS_MAX_NAME_LENGTH	198
5.72.2.27 DNS_MAX_SERVERS	198
5.72.2.28 DNS_MSG_SIZE	198
5.72.2.29 DNS_TABLE_SIZE	198
5.72.2.30 ETH_PAD_SIZE	198
5.72.2.31 ETHARP_DEBUG	198
5.72.2.32 ETHARP_STATS	198
5.72.2.33 ETHARP_SUPPORT_STATIC_ENTRIES	198
5.72.2.34 ETHARP_SUPPORT_VLAN	198
5.72.2.35 ETHARP_TRUST_IP_MAC	199
5.72.2.36 ICMP_DEBUG	199
5.72.2.37 ICMP_STATS	199
5.72.2.38 ICMP_TTL	199
5.72.2.39 IGMP_DEBUG	199
5.72.2.40 IGMP_STATS	199
5.72.2.41 INET_DEBUG	199
5.72.2.42 IP_DEBUG	199
5.72.2.43 IP_DEFAULT_TTL	200
5.72.2.44 IP_FORWARD	200
5.72.2.45 IP_FORWARD_ALLOW_TX_ON_RX_NETIF	200
5.72.2.46 IP_FRAG	200
5.72.2.47 IP_FRAG_USES_STATIC_BUF	200
5.72.2.48 IP_OPTIONS_ALLOWED	200
5.72.2.49 IP_REASS_DEBUG	200
5.72.2.50 IP_REASS_MAX_PBUFS	200
5.72.2.51 IP_REASS_MAXAGE	201
5.72.2.52 IP_REASSEMBLY	201
5.72.2.53 IP_SOF_BROADCAST	201
5.72.2.54 IP_SOF_BROADCAST_RECV	201
5.72.2.55 IP_STATS	201
5.72.2.56 IPFRAG_STATS	201
5.72.2.57 LINK_STATS	201
5.72.2.58 LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT	201
5.72.2.59 LWIP_ARP	202
5.72.2.60 LWIP_AUTOIP	202
5.72.2.61 LWIP_BROADCAST_PING	202
5.72.2.62 LWIP_CALLBACK_API	202
5.72.2.63 LWIP_CHECKSUM_ON_COPY	202
5.72.2.64 LWIP_COMPAT_SOCKETS	202

5.72.2.65 LWIP_DBG_MIN_LEVEL	202
5.72.2.66 LWIP_DBG_TYPES_ON	203
5.72.2.67 LWIP_DHCP	203
5.72.2.68 LWIP_DHCP_AUTOIP_COOP	203
5.72.2.69 LWIP_DHCP_AUTOIP_COOP_TRIES	203
5.72.2.70 LWIP_DNS	203
5.72.2.71 LWIP_ETHERNET	203
5.72.2.72 LWIP_EVENT_API	203
5.72.2.73 LWIP_HAVE_LOOPIF	204
5.72.2.74 LWIP_HAVE_SLIP	204
5.72.2.75 LWIP_ICMP	204
5.72.2.76 LWIP_IGMP	204
5.72.2.77 LWIP_LOOPBACK_MAX_PBUFS	204
5.72.2.78 LWIP_MULTICAST_PING	204
5.72.2.79 LWIP_NETBUF_RECVINFO	204
5.72.2.80 LWIP_NETCONN	204
5.72.2.81 LWIP_NETIF_API	204
5.72.2.82 LWIP_NETIF_HOSTNAME	205
5.72.2.83 LWIP_NETIF_HWADDRHINT	205
5.72.2.84 LWIP_NETIF_LINK_CALLBACK	205
5.72.2.85 LWIP_NETIF_LOOPBACK	205
5.72.2.86 LWIP_NETIF_LOOPBACK_MULTITHREADING	205
5.72.2.87 LWIP_NETIF_REMOVE_CALLBACK	205
5.72.2.88 LWIP_NETIF_STATUS_CALLBACK	205
5.72.2.89 LWIP_NETIF_TX_SINGLE_PBUF	205
5.72.2.90 LWIP_POSIX_SOCKETS_IO_NAMES	206
5.72.2.91 LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS	206
5.72.2.92 LWIP_RAW	206
5.72.2.93 LWIP_SNMP	206
5.72.2.94 LWIP_SO_RCVBUF	206
5.72.2.95 LWIP_SO_RCVTIMEO	206
5.72.2.96 LWIP_SO_SNDTIMEO	206
5.72.2.97 LWIP_SOCKET	206
5.72.2.98 LWIP_STATS	207
5.72.2.99 LWIP_STATS_DISPLAY	207
5.72.2.100 LWIP_TCP	207
5.72.2.101 LWIP_TCP_KEEPALIVE	207
5.72.2.102 LWIP_TCP_TIMESTAMPS	207
5.72.2.103 LWIP_TCPIP_CORE_LOCKING	207
5.72.2.104 LWIP_TCPIP_CORE_LOCKING_INPUT	207

5.72.2.105	WIP_TCPIP_TIMEOUT	207
5.72.2.106	WIP_UDP	207
5.72.2.107	WIP_UDPLITE	208
5.72.2.108	MEM_ALIGNMENT	208
5.72.2.109	MEM_DEBUG	208
5.72.2.110	MEM_LIBC_MALLOC	208
5.72.2.111	MEM_SIZE	208
5.72.2.112	MEM_STATS	208
5.72.2.113	MEM_USE_POOLS	208
5.72.2.114	MEM_USE_POOLS_TRY_BIGGER_POOL	208
5.72.2.115	MEMCPY	209
5.72.2.116	MEMP_DEBUG	209
5.72.2.117	MEMP_MEM_MALLOC	209
5.72.2.118	MEMP_NUM_ARP_QUEUE	209
5.72.2.119	MEMP_NUM_FRAG_PBUF	209
5.72.2.120	MEMP_NUM_IGMP_GROUP	209
5.72.2.121	MEMP_NUM_LOCALHOSTLIST	209
5.72.2.122	MEMP_NUM_NETBUF	209
5.72.2.123	MEMP_NUM_NETCONN	210
5.72.2.124	MEMP_NUM_NETDB	210
5.72.2.125	MEMP_NUM_PBUF	210
5.72.2.126	MEMP_NUM_PPPOE_INTERFACES	210
5.72.2.127	MEMP_NUM_RAW_PCB	210
5.72.2.128	MEMP_NUM_REASSDATA	210
5.72.2.129	MEMP_NUM_SNMP_NODE	210
5.72.2.130	MEMP_NUM_SNMP_ROOTNODE	210
5.72.2.131	MEMP_NUM_SNMP_VALUE	211
5.72.2.132	MEMP_NUM_SNMP_VARBIND	211
5.72.2.133	MEMP_NUM_SYS_TIMEOUT	211
5.72.2.134	MEMP_NUM_TCP_PCB	211
5.72.2.135	MEMP_NUM_TCP_PCB_LISTEN	211
5.72.2.136	MEMP_NUM_TCP_SEG	211
5.72.2.137	MEMP_NUM_TCPIP_MSG_API	211
5.72.2.138	MEMP_NUM_TCPIP_MSG_INPKT	211
5.72.2.139	MEMP_NUM_UDP_PCB	212
5.72.2.140	MEMP_OVERFLOW_CHECK	212
5.72.2.141	MEMP_SANITY_CHECK	212
5.72.2.142	MEMP_SEPARATE_POOLS	212
5.72.2.143	MEMP_STATS	212
5.72.2.144	MEMP_USE_CUSTOM_POOLS	212

5.72.2.145	NETIF_DEBUG	212
5.72.2.146	NO_SYS	212
5.72.2.147	NO_SYS_NO_TIMERS	213
5.72.2.148	PBUF_DEBUG	213
5.72.2.149	PBUF_LINK_HLEN	213
5.72.2.150	PBUF_POOL_BUFSIZE	213
5.72.2.151	PBUF_POOL_SIZE	213
5.72.2.152	PPP_DEBUG	213
5.72.2.153	PPP_SUPPORT	213
5.72.2.154	PPP_THREAD_NAME	213
5.72.2.155	PPP_THREAD_PRIO	213
5.72.2.156	PPP_THREAD_STACKSIZE	214
5.72.2.157	PPPOE_SUPPORT	214
5.72.2.158	PPPOS_SUPPORT	214
5.72.2.159	RAW_DEBUG	214
5.72.2.160	RAW_TTL	214
5.72.2.161	RCV_BUFSIZE_DEFAULT	214
5.72.2.162	SLIP_DEBUG	214
5.72.2.163	SLIIF_THREAD_NAME	214
5.72.2.164	SLIIF_THREAD_PRIO	214
5.72.2.165	SLIIF_THREAD_STACKSIZE	215
5.72.2.166	MEMCPY	215
5.72.2.167	SNMP_CONCURRENT_REQUESTS	215
5.72.2.168	SNMP_MAX_OCTET_STRING_LEN	215
5.72.2.169	SNMP_MAX_TREE_DEPTH	215
5.72.2.170	SNMP_MAX_VALUE_SIZE	215
5.72.2.171	SNMP_MIB_DEBUG	215
5.72.2.172	SNMP_MSG_DEBUG	215
5.72.2.173	SNMP_PRIVATE_MIB	216
5.72.2.174	SNMP_SAFE_REQUESTS	216
5.72.2.175	SNMP_TRAP_DESTINATIONS	216
5.72.2.176	SO_REUSE	216
5.72.2.177	SO_REUSE_RXTOALL	216
5.72.2.178	SOCKETS_DEBUG	216
5.72.2.179	SYS_DEBUG	216
5.72.2.180	SYS_LIGHTWEIGHT_PROT	216
5.72.2.181	SYS_STATS	216
5.72.2.182	TCP_CALCULATE_EFF_SEND_MSS	217
5.72.2.183	TCP_CWND_DEBUG	217
5.72.2.184	TCP_DEBUG	217

5.72.2.185TCP_DEFAULT_LISTEN_BACKLOG	217
5.72.2.186TCP_FR_DEBUG	217
5.72.2.187TCP_INPUT_DEBUG	217
5.72.2.188TCP_LISTEN_BACKLOG	217
5.72.2.189TCP_MAXRTX	217
5.72.2.190TCP_MSS	217
5.72.2.191TCP_OOSEQ_MAX_BYTES	218
5.72.2.192TCP_OOSEQ_MAX_PBUFS	218
5.72.2.193TCP_OUTPUT_DEBUG	218
5.72.2.194TCP_OVERSIZE	218
5.72.2.195TCP_QLEN_DEBUG	218
5.72.2.196TCP_QUEUE_OOSEQ	218
5.72.2.197TCP_RST_DEBUG	218
5.72.2.198TCP_RTO_DEBUG	218
5.72.2.199TCP_SND_BUF	219
5.72.2.200TCP_SND_QUEUELEN	219
5.72.2.201TCP_SNDLOWAT	219
5.72.2.202TCP_SNDQUEUELOWAT	219
5.72.2.203TCP_STATS	219
5.72.2.204TCP_SYNMAXRTX	219
5.72.2.205TCP_TTL	219
5.72.2.206TCP_WND	219
5.72.2.207TCP_WND_DEBUG	220
5.72.2.208TCP_WND_UPDATE_THRESHOLD	220
5.72.2.209TCPIP_DEBUG	220
5.72.2.210TCPIP_MBOX_SIZE	220
5.72.2.211TCPIP_THREAD_NAME	220
5.72.2.212TCPIP_THREAD_PRIO	220
5.72.2.213TCPIP_THREAD_STACKSIZE	220
5.72.2.214TIMERS_DEBUG	220
5.72.2.215UDP_DEBUG	220
5.72.2.216UDP_STATS	221
5.72.2.217UDP_TTL	221
5.73 src/include/lwip/pbuf.h File Reference	221
5.73.1 Macro Definition Documentation	223
5.73.1.1 LWIP_SUPPORT_CUSTOM_PBUF	223
5.73.1.2 PBUF_FLAG_IS_CUSTOM	223
5.73.1.3 PBUF_FLAG_LLBCAST	223
5.73.1.4 PBUF_FLAG_LLMCAST	223
5.73.1.5 PBUF_FLAG_MCASTLOOP	223

5.73.1.6	PBUF_FLAG_PUSH	223
5.73.1.7	PBUF_FLAG_TCP_FIN	223
5.73.1.8	pbuf_init	223
5.73.1.9	PBUF_IP_HLEN	223
5.73.1.10	PBUF_TRANSPORT_HLEN	224
5.73.2	Enumeration Type Documentation	224
5.73.2.1	pbuf_layer	224
5.73.2.2	pbuf_type	224
5.73.3	Function Documentation	224
5.73.3.1	pbuf_alloc	224
5.73.3.2	pbuf_cat	225
5.73.3.3	pbuf_chain	225
5.73.3.4	pbuf_clen	225
5.73.3.5	pbuf_coalesce	226
5.73.3.6	pbuf_copy	226
5.73.3.7	pbuf_copy_partial	226
5.73.3.8	pbuf_dechain	227
5.73.3.9	pbuf_free	227
5.73.3.10	pbuf_get_at	227
5.73.3.11	pbuf_header	228
5.73.3.12	pbuf_memcmp	228
5.73.3.13	pbuf_memfind	228
5.73.3.14	pbuf_realloc	228
5.73.3.15	pbuf_ref	229
5.73.3.16	pbuf_strstr	229
5.73.3.17	pbuf_take	229
5.74	src/include/lwip/raw.h File Reference	230
5.75	src/include/lwip/sio.h File Reference	230
5.75.1	Typedef Documentation	231
5.75.1.1	sio_fd_t	231
5.75.2	Function Documentation	231
5.75.2.1	sio_open	231
5.75.2.2	sio_read	232
5.75.2.3	sio_read_abort	233
5.75.2.4	sio_recv	233
5.75.2.5	sio_send	233
5.75.2.6	sio_tryread	233
5.75.2.7	sio_write	234
5.76	src/include/lwip/snmp.h File Reference	234
5.76.1	Macro Definition Documentation	238

5.76.1.1	snmp_add_ifinoctets	238
5.76.1.2	snmp_add_ifoutoctets	238
5.76.1.3	snmp_add_snmpintotalreqvars	238
5.76.1.4	snmp_add_snmpintotalsetvars	238
5.76.1.5	snmp_add_sysuptime	238
5.76.1.6	snmp_dec_iflist	238
5.76.1.7	snmp_delete_arpidx_tree	238
5.76.1.8	snmp_delete_ipaddridx_tree	238
5.76.1.9	snmp_delete_iprtidx_tree	238
5.76.1.10	snmp_delete_udpidx_tree	239
5.76.1.11	snmp_get_snmpenableauthentraps	239
5.76.1.12	snmp_get_snmpgrpid_ptr	239
5.76.1.13	snmp_get_sysobjid_ptr	239
5.76.1.14	snmp_get_sysuptime	239
5.76.1.15	snmp_inc_icmpinaddrmaskreps	239
5.76.1.16	snmp_inc_icmpinaddrmasks	239
5.76.1.17	snmp_inc_icmpindestunreachs	239
5.76.1.18	snmp_inc_icmpinechoreps	239
5.76.1.19	snmp_inc_icmpinechos	239
5.76.1.20	snmp_inc_icmpinerrors	239
5.76.1.21	snmp_inc_icmpinmsgs	239
5.76.1.22	snmp_inc_icmpinparmprobs	240
5.76.1.23	snmp_inc_icmpinredirects	240
5.76.1.24	snmp_inc_icmpinsrcquenchs	240
5.76.1.25	snmp_inc_icmpintimeexcds	240
5.76.1.26	snmp_inc_icmpintimestampreps	240
5.76.1.27	snmp_inc_icmpintimestamps	240
5.76.1.28	snmp_inc_icmpoutaddrmaskreps	240
5.76.1.29	snmp_inc_icmpoutaddrmasks	240
5.76.1.30	snmp_inc_icmpoutdestunreachs	240
5.76.1.31	snmp_inc_icmpoutechoreps	240
5.76.1.32	snmp_inc_icmpoutechos	240
5.76.1.33	snmp_inc_icmpouterrors	240
5.76.1.34	snmp_inc_icmpoutmsgs	241
5.76.1.35	snmp_inc_icmpoutparmprobs	241
5.76.1.36	snmp_inc_icmpoutredirects	241
5.76.1.37	snmp_inc_icmpoutsrcquenchs	241
5.76.1.38	snmp_inc_icmpouttimeexcds	241
5.76.1.39	snmp_inc_icmpouttimestampreps	241
5.76.1.40	snmp_inc_icmpouttimestamps	241

5.76.1.41 snmp_inc_ifindiscards	241
5.76.1.42 snmp_inc_ifinnucastpkts	241
5.76.1.43 snmp_inc_ifinucastpkts	241
5.76.1.44 snmp_inc_iflist	241
5.76.1.45 snmp_inc_ifoutdiscards	241
5.76.1.46 snmp_inc_ifoutnucastpkts	242
5.76.1.47 snmp_inc_ifoutucastpkts	242
5.76.1.48 snmp_inc_ipforwdatagrams	242
5.76.1.49 snmp_inc_ipfragcreates	242
5.76.1.50 snmp_inc_ipfragfails	242
5.76.1.51 snmp_inc_ipfragoks	242
5.76.1.52 snmp_inc_ipinaddrerrors	242
5.76.1.53 snmp_inc_ipindelivers	242
5.76.1.54 snmp_inc_ipindiscards	242
5.76.1.55 snmp_inc_ipinhdrrerrors	242
5.76.1.56 snmp_inc_ipinreceives	242
5.76.1.57 snmp_inc_ipinunknownprotos	242
5.76.1.58 snmp_inc_ipoutdiscards	243
5.76.1.59 snmp_inc_ipoutnoroutes	243
5.76.1.60 snmp_inc_ipoutrequests	243
5.76.1.61 snmp_inc_ippreasmfails	243
5.76.1.62 snmp_inc_ippreasmoks	243
5.76.1.63 snmp_inc_ippreasmreqds	243
5.76.1.64 snmp_inc_iproutingdiscards	243
5.76.1.65 snmp_inc_snmpinasnparseerrs	243
5.76.1.66 snmp_inc_snmpinbadcommunitynames	243
5.76.1.67 snmp_inc_snmpinbadcommunityuses	243
5.76.1.68 snmp_inc_snmpinbadvalues	243
5.76.1.69 snmp_inc_snmpinbadversions	243
5.76.1.70 snmp_inc_snmpingenerrs	244
5.76.1.71 snmp_inc_snmpingetnexts	244
5.76.1.72 snmp_inc_snmpingetrequests	244
5.76.1.73 snmp_inc_snmpingetresponses	244
5.76.1.74 snmp_inc_snmpinnosuchnames	244
5.76.1.75 snmp_inc_snmpinpkts	244
5.76.1.76 snmp_inc_snmpinreadonlys	244
5.76.1.77 snmp_inc_snmpinsetrequests	244
5.76.1.78 snmp_inc_snmpintoobigs	244
5.76.1.79 snmp_inc_snmpintraps	244
5.76.1.80 snmp_inc_snmpoutbadvalues	244

5.76.1.81	snmp_inc_snmpoutgenerrs	244
5.76.1.82	snmp_inc_snmpoutgetnexts	245
5.76.1.83	snmp_inc_snmpoutgetrequests	245
5.76.1.84	snmp_inc_snmpoutgetresponses	245
5.76.1.85	snmp_inc_snmpoutnosuchnames	245
5.76.1.86	snmp_inc_snmpoutpkts	245
5.76.1.87	snmp_inc_snmpoutsetrequests	245
5.76.1.88	snmp_inc_snmpouttoobigs	245
5.76.1.89	snmp_inc_snmpouttraps	245
5.76.1.90	snmp_inc_sysuptime	245
5.76.1.91	snmp_inc_tcpactiveopens	245
5.76.1.92	snmp_inc_tcpattemptfails	245
5.76.1.93	snmp_inc_tcestabresets	245
5.76.1.94	snmp_inc_tcpinerrs	246
5.76.1.95	snmp_inc_tcpinsegs	246
5.76.1.96	snmp_inc_tcpoutrsts	246
5.76.1.97	snmp_inc_tcpoutsegs	246
5.76.1.98	snmp_inc_tcppassiveopens	246
5.76.1.99	snmp_inc_tcpretranssegs	246
5.76.1.100	snmp_inc_udpindatagrams	246
5.76.1.101	snmp_inc_udpinerrors	246
5.76.1.102	snmp_inc_udpnoports	246
5.76.1.103	snmp_inc_udpoutdatagrams	246
5.76.1.104	snmp_insert_arpidx_tree	246
5.76.1.105	snmp_insert_ipaddridx_tree	246
5.76.1.106	snmp_insert_iprteidx_tree	247
5.76.1.107	snmp_insert_udpidx_tree	247
5.76.1.108	snmp_set_snmpenableauthentraps	247
5.76.1.109	snmp_set_syscontact	247
5.76.1.110	snmp_set_sysdescr	247
5.76.1.111	snmp_set_syslocation	247
5.76.1.112	snmp_set_sysname	247
5.76.1.113	snmp_set_sysobjid	247
5.76.2	Enumeration Type Documentation	247
5.76.2.1	snmp_ifType	247
5.77	src/include/lwip/snmp_asn1.h File Reference	248
5.77.1	Detailed Description	249
5.78	src/include/lwip/snmp_msg.h File Reference	249
5.78.1	Detailed Description	250
5.79	src/include/lwip/snmp_structs.h File Reference	251

5.79.1 Detailed Description	252
5.80 src/include/lwip/sockets.h File Reference	252
5.81 src/include/lwip/stats.h File Reference	253
5.81.1 Macro Definition Documentation	254
5.81.1.1 ETHARP_STATS_DISPLAY	254
5.81.1.2 ETHARP_STATS_INC	254
5.81.1.3 ICMP_STATS_DISPLAY	254
5.81.1.4 ICMP_STATS_INC	254
5.81.1.5 IGMP_STATS_DISPLAY	255
5.81.1.6 IGMP_STATS_INC	255
5.81.1.7 IP_STATS_DISPLAY	255
5.81.1.8 IP_STATS_INC	255
5.81.1.9 IPFRAG_STATS_DISPLAY	255
5.81.1.10 IPFRAG_STATS_INC	255
5.81.1.11 LINK_STATS_DISPLAY	255
5.81.1.12 LINK_STATS_INC	255
5.81.1.13 MEM_STATS_AVAIL	255
5.81.1.14 MEM_STATS_DEC_USED	255
5.81.1.15 MEM_STATS_DISPLAY	255
5.81.1.16 MEM_STATS_INC	255
5.81.1.17 MEM_STATS_INC_USED	256
5.81.1.18 MEMP_STATS_AVAIL	256
5.81.1.19 MEMP_STATS_DEC	256
5.81.1.20 MEMP_STATS_DISPLAY	256
5.81.1.21 MEMP_STATS_INC	256
5.81.1.22 MEMP_STATS_INC_USED	256
5.81.1.23 STATS_DEC	256
5.81.1.24 stats_display	256
5.81.1.25 stats_display_igmp	256
5.81.1.26 stats_display_mem	256
5.81.1.27 stats_display_memp	256
5.81.1.28 stats_display_proto	256
5.81.1.29 stats_display_sys	257
5.81.1.30 STATS_INC	257
5.81.1.31 STATS_INC_USED	257
5.81.1.32 stats_init	257
5.81.1.33 SYS_STATS_DEC	257
5.81.1.34 SYS_STATS_DISPLAY	257
5.81.1.35 SYS_STATS_INC	257
5.81.1.36 SYS_STATS_INC_USED	257

5.81.1.37	TCP_STATS_DISPLAY	257
5.81.1.38	TCP_STATS_INC	257
5.81.1.39	UDP_STATS_DISPLAY	257
5.81.1.40	UDP_STATS_INC	257
5.82	src/include/lwip/sys.h File Reference	258
5.82.1	Macro Definition Documentation	259
5.82.1.1	SYS_ARCH_DEC	259
5.82.1.2	SYS_ARCH_DECL_PROTECT	260
5.82.1.3	SYS_ARCH_GET	260
5.82.1.4	SYS_ARCH_INC	260
5.82.1.5	SYS_ARCH_PROTECT	260
5.82.1.6	SYS_ARCH_SET	260
5.82.1.7	SYS_ARCH_TIMEOUT	261
5.82.1.8	SYS_ARCH_UNPROTECT	261
5.82.1.9	SYS_MBOX_EMPTY	261
5.82.1.10	sys_mbox_fetch	261
5.82.1.11	sys_mbox_tryfetch	261
5.82.1.12	sys_sem_wait	261
5.82.2	Typedef Documentation	261
5.82.2.1	lwip_thread_fn	261
5.82.3	Function Documentation	261
5.82.3.1	sys_arch_mbox_fetch	261
5.82.3.2	sys_arch_mbox_tryfetch	262
5.82.3.3	sys_arch_sem_wait	262
5.82.3.4	sys_init	262
5.82.3.5	sys_jiffies	262
5.82.3.6	sys_mbox_free	262
5.82.3.7	sys_mbox_new	262
5.82.3.8	sys_mbox_post	263
5.82.3.9	sys_mbox_set_invalid	263
5.82.3.10	sys_mbox_trypost	263
5.82.3.11	sys_mbox_valid	263
5.82.3.12	sys_msleep	263
5.82.3.13	sys_mutex_free	263
5.82.3.14	sys_mutex_lock	264
5.82.3.15	sys_mutex_new	264
5.82.3.16	sys_mutex_set_invalid	264
5.82.3.17	sys_mutex_unlock	264
5.82.3.18	sys_mutex_valid	264
5.82.3.19	sys_now	264

5.82.3.20	sys_sem_free	264
5.82.3.21	sys_sem_new	265
5.82.3.22	sys_sem_set_invalid	265
5.82.3.23	sys_sem_signal	265
5.82.3.24	sys_sem_valid	265
5.82.3.25	sys_thread_new	265
5.83	src/include/lwip/tcp.h File Reference	266
5.84	src/include/lwip/tcp_impl.h File Reference	267
5.85	src/include/lwip/tcpip.h File Reference	267
5.85.1	Macro Definition Documentation	269
5.85.1.1	LOCK_TCPIP_CORE	269
5.85.1.2	LWIP_TCPIP_THREAD_ALIVE	269
5.85.1.3	TCPIP_APIMSG	269
5.85.1.4	TCPIP_APIMSG_ACK	269
5.85.1.5	tcpip_callback	269
5.85.1.6	TCPIP_NETIFAPI	269
5.85.1.7	TCPIP_NETIFAPI_ACK	269
5.85.1.8	UNLOCK_TCPIP_CORE	270
5.85.2	Typedef Documentation	270
5.85.2.1	tcpip_callback_fn	270
5.85.2.2	tcpip_init_done_fn	270
5.85.3	Enumeration Type Documentation	270
5.85.3.1	tcpip_msg_type	270
5.85.4	Function Documentation	270
5.85.4.1	mem_free_callback	270
5.85.4.2	pbuf_free_callback	270
5.85.4.3	tcpip_callback_with_block	271
5.85.4.4	tcpip_callbackmsg_delete	271
5.85.4.5	tcpip_callbackmsg_new	271
5.85.4.6	tcpip_init	271
5.85.4.7	tcpip_input	272
5.85.4.8	tcpip_trycallback	272
5.86	src/include/lwip/timers.h File Reference	272
5.86.1	Macro Definition Documentation	274
5.86.1.1	LWIP_DEBUG_TIMERNAMES	274
5.86.1.2	LWIP_TIMERS	274
5.86.2	Typedef Documentation	274
5.86.2.1	sys_timeout_handler	274
5.86.3	Function Documentation	274
5.86.3.1	sys_timeout	274

5.86.3.2	sys_timeouts_init	274
5.86.3.3	sys_timeouts_mbox_fetch	274
5.86.3.4	sys_untimeout	274
5.87	src/include/lwip/udp.h File Reference	275
5.88	src/include/netif/etharp.h File Reference	275
5.89	src/include/netif/ppp_oe.h File Reference	276
5.90	src/include/netif/slipif.h File Reference	277
5.90.1	Macro Definition Documentation	279
5.90.1.1	SLIP_RX_FROM_ISR	279
5.90.1.2	SLIP_RX_QUEUE	279
5.90.1.3	SLIP_USE_RX_THREAD	279
5.90.2	Function Documentation	279
5.90.2.1	slipif_init	279
5.90.2.2	slipif_poll	279
5.91	src/include/posix/sys/socket.h File Reference	279
5.91.1	Detailed Description	280
5.92	src/netif/etharp.c File Reference	281
5.92.1	Detailed Description	281
5.93	src/netif/ethernetif.c File Reference	281
5.93.1	Detailed Description	282
5.94	src/netif/ppp/auth.c File Reference	283
5.95	src/netif/ppp/auth.h File Reference	283
5.95.1	Function Documentation	284
5.95.1.1	auth_check_options	284
5.95.1.2	auth_ip_addr	284
5.95.1.3	auth_peer_fail	284
5.95.1.4	auth_peer_success	284
5.95.1.5	auth_reset	284
5.95.1.6	auth_withpeer_fail	284
5.95.1.7	auth_withpeer_success	284
5.95.1.8	bad_ip_adrs	284
5.95.1.9	check_passwd	284
5.95.1.10	get_secret	284
5.95.1.11	link_down	284
5.95.1.12	link_established	284
5.95.1.13	link_required	284
5.95.1.14	link_terminated	284
5.95.1.15	np_down	284
5.95.1.16	np_finished	284
5.95.1.17	np_up	284

5.96	src/netif/ppp/chap.c File Reference	284
5.97	src/netif/ppp/chap.h File Reference	285
5.97.1	Macro Definition Documentation	286
5.97.1.1	CHAP_CHALLENGE	286
5.97.1.2	CHAP_DIGEST_MD5	286
5.97.1.3	CHAP_FAILURE	286
5.97.1.4	CHAP_HEADERLEN	286
5.97.1.5	CHAP_MICROSOFT	286
5.97.1.6	CHAP_RESPONSE	286
5.97.1.7	CHAP_SUCCESS	287
5.97.1.8	CHAPCS_CLOSED	287
5.97.1.9	CHAPCS_INITIAL	287
5.97.1.10	CHAPCS_LISTEN	287
5.97.1.11	CHAPCS_OPEN	287
5.97.1.12	CHAPCS_PENDING	287
5.97.1.13	CHAPCS_RESPONSE	287
5.97.1.14	CHAPSS_BADAUTH	287
5.97.1.15	CHAPSS_CLOSED	287
5.97.1.16	CHAPSS_INITIAL	287
5.97.1.17	CHAPSS_INITIAL_CHAL	287
5.97.1.18	CHAPSS_OPEN	287
5.97.1.19	CHAPSS_PENDING	288
5.97.1.20	CHAPSS_RECHALLENGE	288
5.97.1.21	MAX_CHALLENGE_LENGTH	288
5.97.1.22	MAX_RESPONSE_LENGTH	288
5.97.1.23	MD5_SIGNATURE_SIZE	288
5.97.1.24	MIN_CHALLENGE_LENGTH	288
5.97.1.25	MS_CHAP_RESPONSE_LEN	288
5.97.2	Typedef Documentation	288
5.97.2.1	chap_state	288
5.97.3	Function Documentation	288
5.97.3.1	ChapAuthPeer	288
5.97.3.2	ChapAuthWithPeer	288
5.97.4	Variable Documentation	288
5.97.4.1	chap	288
5.97.4.2	chap_protent	288
5.98	src/netif/ppp/chpms.c File Reference	289
5.98.1	Macro Definition Documentation	289
5.98.1.1	USE_CRYPT	289
5.99	src/netif/ppp/chpms.h File Reference	289

5.99.1	Macro Definition Documentation	290
5.99.1.1	MAX_NT_PASSWORD	290
5.99.2	Function Documentation	290
5.99.2.1	ChapMS	290
5.100	src/netif/ppp/fsm.c File Reference	290
5.101	src/netif/ppp/fsm.h File Reference	290
5.101.1	Macro Definition Documentation	291
5.101.1.1	CODEREJ	291
5.101.1.2	CONFACK	291
5.101.1.3	CONFNAK	292
5.101.1.4	CONFREJ	292
5.101.1.5	CONFREQ	292
5.101.1.6	HEADERLEN	292
5.101.1.7	LS_ACKRCVD	292
5.101.1.8	LS_ACKSENT	292
5.101.1.9	LS_CLOSED	292
5.101.1.10	LS_CLOSING	292
5.101.1.11	LS_INITIAL	292
5.101.1.12	LS_OPENED	292
5.101.1.13	LS_REQSENT	292
5.101.1.14	LS_STARTING	292
5.101.1.15	LS_STOPPED	293
5.101.1.16	LS_STOPPING	293
5.101.1.17	OPT_PASSIVE	293
5.101.1.18	OPT_RESTART	293
5.101.1.19	OPT_SILENT	293
5.101.1.20	TERMACK	293
5.101.1.21	TERMREQ	293
5.101.2	Typedef Documentation	293
5.101.2.1	fsm	293
5.101.2.2	fsm_callbacks	293
5.101.3	Function Documentation	293
5.101.3.1	fsm_close	293
5.101.3.2	fsm_init	293
5.101.3.3	fsm_input	293
5.101.3.4	fsm_lowerdown	293
5.101.3.5	fsm_lowerup	293
5.101.3.6	fsm_open	293
5.101.3.7	fsm_protreject	293
5.101.3.8	fsm_sdata	293

5.101.4 Variable Documentation	294
5.101.4.1 peer_mru	294
5.102src/netif/ppp/ipcp.c File Reference	294
5.103src/netif/ppp/ipcp.h File Reference	294
5.103.1 Macro Definition Documentation	295
5.103.1.1 CI_ADDR	295
5.103.1.2 CI_ADDRS	295
5.103.1.3 CI_COMPRESSTYPE	295
5.103.1.4 CI_MS_DNS1	295
5.103.1.5 CI_MS_DNS2	295
5.103.1.6 CI_MS_WINS1	295
5.103.1.7 CI_MS_WINS2	295
5.103.1.8 IPCP_VJ_COMP	296
5.103.1.9 IPCP_VJ_COMP_OLD	296
5.103.1.10PCP_VJMODE_OLD	296
5.103.1.11PCP_VJMODE_RFC1172	296
5.103.1.12PCP_VJMODE_RFC1332	296
5.103.2 Typedef Documentation	296
5.103.2.1 ipcp_options	296
5.103.3 Variable Documentation	296
5.103.3.1 ipcp_allowoptions	296
5.103.3.2 ipcp_fsm	296
5.103.3.3 ipcp_gotoptions	296
5.103.3.4 ipcp_hisoptions	296
5.103.3.5 ipcp_protent	296
5.103.3.6 ipcp_wantoptions	296
5.104src/netif/ppp/lcp.c File Reference	297
5.105src/netif/ppp/lcp.h File Reference	297
5.105.1 Macro Definition Documentation	298
5.105.1.1 CBCP_OPT	298
5.105.1.2 CI_ACCOMPRESSSION	298
5.105.1.3 CI_ASYNCMAP	298
5.105.1.4 CI_AUTHTYPE	299
5.105.1.5 CI_CALLBACK	299
5.105.1.6 CI_EPDISC	299
5.105.1.7 CI_MAGICNUMBER	299
5.105.1.8 CI_MRRU	299
5.105.1.9 CI_MRU	299
5.105.1.10CI_PCOMPRESSION	299
5.105.1.11CI_QUALITY	299

5.105.1.12	CI_SSNHF	299
5.105.1.13	DEFLOOPBACKFAIL	299
5.105.1.14	DISCREQ	299
5.105.1.15	ECHOREP	299
5.105.1.16	ECHOREQ	300
5.105.1.17	PROTREJ	300
5.105.2	Typedef Documentation	300
5.105.2.1	lcp_options	300
5.105.3	Enumeration Type Documentation	300
5.105.3.1	LinkPhase	300
5.105.4	Function Documentation	300
5.105.4.1	lcp_close	300
5.105.4.2	lcp_init	300
5.105.4.3	lcp_lowerdown	300
5.105.4.4	lcp_lowerup	300
5.105.4.5	lcp_open	300
5.105.4.6	lcp_sprotrej	300
5.105.5	Variable Documentation	300
5.105.5.1	lcp_allowoptions	300
5.105.5.2	lcp_gotoptions	300
5.105.5.3	lcp_hisoptions	300
5.105.5.4	lcp_phase	300
5.105.5.5	lcp_protent	300
5.105.5.6	lcp_wantoptions	301
5.105.5.7	xmit_accm	301
5.106	src/netif/ppp/magic.c File Reference	301
5.107	src/netif/ppp/magic.h File Reference	301
5.107.1	Function Documentation	301
5.107.1.1	magic	301
5.107.1.2	magicInit	301
5.108	src/netif/ppp/md5.c File Reference	301
5.109	src/netif/ppp/md5.h File Reference	302
5.109.1	Function Documentation	302
5.109.1.1	MD5Final	302
5.109.1.2	MD5Init	302
5.109.1.3	MD5Update	302
5.110	src/netif/ppp/pap.c File Reference	302
5.111	src/netif/ppp/pap.h File Reference	303
5.112	src/netif/ppp/ppp.c File Reference	303
5.113	src/netif/ppp/ppp.h File Reference	304

5.114src/netif/ppp/ppp_impl.h File Reference	305
5.115src/netif/ppp/ppp_oe.c File Reference	306
5.116src/netif/ppp/pppdebug.h File Reference	307
5.116.1 Macro Definition Documentation	307
5.116.1.1 AUTHDEBUG	307
5.116.1.2 CHAPDEBUG	308
5.116.1.3 FSMDEBUG	308
5.116.1.4 IPCPDEBUG	308
5.116.1.5 LCPDEBUG	308
5.116.1.6 LOG_CRITICAL	308
5.116.1.7 LOG_DEBUG	308
5.116.1.8 LOG_DETAIL	308
5.116.1.9 LOG_ERR	308
5.116.1.10LOG_INFO	308
5.116.1.11LOG_NOTICE	308
5.116.1.12LOG_WARNING	308
5.116.1.13PPPDEBUG	308
5.116.1.14TRACELCP	309
5.116.1.15JPAPDEBUG	309
5.117src/netif/ppp/randm.c File Reference	309
5.118src/netif/ppp/randm.h File Reference	309
5.118.1 Function Documentation	310
5.118.1.1 avChurnRand	310
5.118.1.2 avGenRand	310
5.118.1.3 avRandom	310
5.118.1.4 avRandomInit	310
5.118.1.5 avRandomize	310
5.119src/netif/ppp/vj.c File Reference	310
5.120src/netif/ppp/vj.h File Reference	310
5.120.1 Macro Definition Documentation	312
5.120.1.1 cs_hdr	312
5.120.1.2 cs_ip	312
5.120.1.3 MAX_HDR	312
5.120.1.4 MAX_SLOTS	312
5.120.1.5 NEW_A	312
5.120.1.6 NEW_C	312
5.120.1.7 NEW_I	312
5.120.1.8 NEW_S	312
5.120.1.9 NEW_U	313
5.120.1.10NEW_W	313

5.120.1.11	SPECIAL_D	313
5.120.1.12	SPECIAL_I	313
5.120.1.13	SPECIALS_MASK	313
5.120.1.14	TCP_PUSH_BIT	313
5.120.1.15	TYPE_COMPRESSED_TCP	313
5.120.1.16	TYPE_ERROR	313
5.120.1.17	TYPE_IP	313
5.120.1.18	TYPE_UNCOMPRESSED_TCP	313
5.120.1.19	VJF_TOSS	313
5.120.2	Function Documentation	313
5.120.2.1	vj_compress_init	313
5.120.2.2	vj_compress_tcp	313
5.120.2.3	vj_uncompress_err	314
5.120.2.4	vj_uncompress_tcp	314
5.120.2.5	vj_uncompress_uncomp	314
5.121	src/netif/slipif.c File Reference	314
5.121.1	Detailed Description	315
Index		317

Chapter 1

Todo List

File [asn1_dec.c](#)

not optimised (yet), favor correctness over speed, favor speed over size

File [asn1_enc.c](#)

not optimised (yet), favor correctness over speed, favor speed over size

globalScope> Global [LWIP_NETIF_TX_SINGLE_PBUF](#)

: TCP and IP-frag do not work with this, yet:

File [snmp_structs.h](#)

namespace prefixes

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

chap_state	9
cstate	11
fsm	13
fsm_callbacks	15
icmp_echo_hdr	17
in_addr	17
ip_addr	18
ip_addr2	18
ip_addr_packed	19
ip_hdr	19
ip_pcb	22
ipcp_options	22
lcp_options	24
MD5_CTX	26
mem	27
memp	28
netbuf	29
netif	30
pbuf	32
sys_timeo	33
tcpip_msg	34
vjcompress	36
vjstat	38

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

src/api/api_lib.c	41
src/api/api_msg.c	42
src/api/err.c	43
src/api/netbuf.c	44
src/api/netdb.c	45
src/api/netifapi.c	46
src/api/sockets.c	47
src/api/tcpip.c	47
src/core/def.c	50
src/core/dhcp.c	53
src/core/dns.c	54
src/core/init.c	55
src/core/mem.c	76
src/core/memp.c	79
src/core/netif.c	82
src/core/pbuf.c	87
src/core/raw.c	94
src/core/stats.c	101
src/core/sys.c	102
src/core/tcp.c	104
src/core/tcp_in.c	104
src/core/tcp_out.c	105
src/core/timers.c	106
src/core/udp.c	108
src/core/ipv4/autoip.c	56
src/core/ipv4/icmp.c	57
src/core/ipv4/igmp.c	58
src/core/ipv4/inet.c	58
src/core/ipv4/inet_chksum.c	59
src/core/ipv4/ip.c	62
src/core/ipv4/ip_addr.c	65
src/core/ipv4/ip_frag.c	70
src/core/ipv6/icmp6.c	71
src/core/ipv6/inet6.c	71
src/core/ipv6/ip6.c	73
src/core/ipv6/ip6_addr.c	74
src/core/snmp/asn1_dec.c	96
src/core/snmp/asn1_enc.c	96

src/core/snmp/mib2.c	97
src/core/snmp/mib_structs.c	98
src/core/snmp/msg_in.c	100
src/core/snmp/msg_out.c	101
src/include/ipv4/lwip/autoip.h	109
src/include/ipv4/lwip/icmp.h	110
src/include/ipv4/lwip/igmp.h	114
src/include/ipv4/lwip/inet.h	116
src/include/ipv4/lwip/inet_chksum.h	122
src/include/ipv4/lwip/ip.h	125
src/include/ipv4/lwip/ip_addr.h	138
src/include/ipv4/lwip/ip_frag.h	151
src/include/ipv6/lwip/icmp.h	113
src/include/ipv6/lwip/inet.h	120
src/include/ipv6/lwip/ip.h	134
src/include/ipv6/lwip/ip_addr.h	149
src/include/lwip/api.h	153
src/include/lwip/api_msg.h	154
src/include/lwip/arch.h	155
src/include/lwip/debug.h	156
src/include/lwip/def.h	159
src/include/lwip/dhcp.h	162
src/include/lwip/dns.h	162
src/include/lwip/err.h	163
src/include/lwip/init.h	166
src/include/lwip/mem.h	169
src/include/lwip/memp.h	173
src/include/lwip/memp_std.h	175
src/include/lwip/netbuf.h	176
src/include/lwip/netdb.h	179
src/include/lwip/netif.h	180
src/include/lwip/netifapi.h	189
src/include/lwip/opt.h	190
src/include/lwip/pbuf.h	221
src/include/lwip/raw.h	230
src/include/lwip/sio.h	230
src/include/lwip/snmp.h	234
src/include/lwip/snmp_asn1.h	248
src/include/lwip/snmp_msg.h	249
src/include/lwip/snmp_structs.h	251
src/include/lwip/sockets.h	252
src/include/lwip/stats.h	253
src/include/lwip/sys.h	258
src/include/lwip/tcp.h	266
src/include/lwip/tcp_impl.h	267
src/include/lwip/tcpip.h	267
src/include/lwip/timers.h	272
src/include/lwip/udp.h	275
src/include/netif/etharp.h	275
src/include/netif/ppp_oe.h	276
src/include/netif/slipif.h	277
src/include/posix/netdb.h	179
src/include/posix/sys/socket.h	279
src/netif/etharp.c	281
src/netif/ethernetif.c	281
src/netif/slipif.c	314
src/netif/ppp/auth.c	283
src/netif/ppp/auth.h	283

src/netif/ppp/chap.c	284
src/netif/ppp/chap.h	285
src/netif/ppp/chpms.c	289
src/netif/ppp/chpms.h	289
src/netif/ppp/fsm.c	290
src/netif/ppp/fsm.h	290
src/netif/ppp/ipcp.c	294
src/netif/ppp/ipcp.h	294
src/netif/ppp/lcp.c	297
src/netif/ppp/lcp.h	297
src/netif/ppp/magic.c	301
src/netif/ppp/magic.h	301
src/netif/ppp/md5.c	301
src/netif/ppp/md5.h	302
src/netif/ppp/pap.c	302
src/netif/ppp/pap.h	303
src/netif/ppp/ppp.c	303
src/netif/ppp/ppp.h	304
src/netif/ppp/ppp_impl.h	305
src/netif/ppp/ppp_oe.c	306
src/netif/ppp/pppdebug.h	307
src/netif/ppp/randm.c	309
src/netif/ppp/randm.h	309
src/netif/ppp/vj.c	310
src/netif/ppp/vj.h	310

Chapter 4

Data Structure Documentation

4.1 chap_state Struct Reference

```
#include <chap.h>
```

Data Fields

- int [unit](#)
- int [clientstate](#)
- int [serverstate](#)
- u_char [challenge](#) [MAX_CHALLENGE_LENGTH]
- u_char [chal_len](#)
- u_char [chal_id](#)
- u_char [chal_type](#)
- u_char [id](#)
- char * [chal_name](#)
- int [chal_interval](#)
- int [timeouttime](#)
- int [max_transmits](#)
- int [chal_transmits](#)
- int [resp_transmits](#)
- u_char [response](#) [MAX_RESPONSE_LENGTH]
- u_char [resp_length](#)
- u_char [resp_id](#)
- u_char [resp_type](#)
- char * [resp_name](#)

4.1.1 Detailed Description

Definition at line 99 of file chap.h.

4.1.2 Field Documentation

4.1.2.1 u_char chal_id

Definition at line 105 of file chap.h.

4.1.2.2 int chal_interval

Definition at line 109 of file chap.h.

4.1.2.3 u_char chal_len

Definition at line 104 of file chap.h.

4.1.2.4 char* chal_name

Definition at line 108 of file chap.h.

4.1.2.5 int chal_transmits

Definition at line 112 of file chap.h.

4.1.2.6 u_char chal_type

Definition at line 106 of file chap.h.

4.1.2.7 u_char challenge[MAX_CHALLENGE_LENGTH]

Definition at line 103 of file chap.h.

4.1.2.8 int clientstate

Definition at line 101 of file chap.h.

4.1.2.9 u_char id

Definition at line 107 of file chap.h.

4.1.2.10 int max_transmits

Definition at line 111 of file chap.h.

4.1.2.11 u_char resp_id

Definition at line 116 of file chap.h.

4.1.2.12 u_char resp_length

Definition at line 115 of file chap.h.

4.1.2.13 char* resp_name

Definition at line 118 of file chap.h.

4.1.2.14 int resp_transmits

Definition at line 113 of file chap.h.

4.1.2.15 u_char resp_type

Definition at line 117 of file chap.h.

4.1.2.16 u_char response[MAX_RESPONSE_LENGTH]

Definition at line 114 of file chap.h.

4.1.2.17 int serverstate

Definition at line 102 of file chap.h.

4.1.2.18 int timeouttime

Definition at line 110 of file chap.h.

4.1.2.19 int unit

Definition at line 100 of file chap.h.

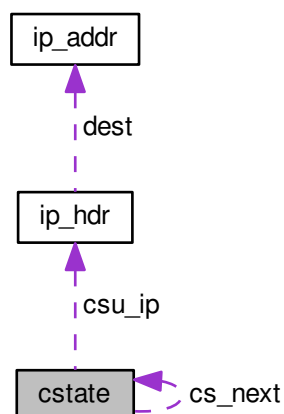
The documentation for this struct was generated from the following file:

- [src/netif/ppp/chap.h](#)

4.2 cstate Struct Reference

```
#include <vj.h>
```

Collaboration diagram for cstate:



Data Fields

- struct `cstate` * `cs_next`
- u_short `cs_hlen`
- u_char `cs_id`
- u_char `cs_filler`
- union {
 - char `csu_hdr` [MAX_HDR]
 - struct `ip_hdr` `csu_ip`
- } `vjcs_u`

4.2.1 Detailed Description

Definition at line 105 of file `vj.h`.

4.2.2 Field Documentation

4.2.2.1 u_char cs_filler

Definition at line 109 of file `vj.h`.

4.2.2.2 u_short cs_hlen

Definition at line 107 of file `vj.h`.

4.2.2.3 u_char cs_id

Definition at line 108 of file `vj.h`.

4.2.2.4 struct cstate* cs_next

Definition at line 106 of file `vj.h`.

4.2.2.5 char csu_hdr[MAX_HDR]

Definition at line 111 of file `vj.h`.

4.2.2.6 struct ip_hdr csu_ip

Definition at line 112 of file `vj.h`.

4.2.2.7 union { ... } vjcs_u

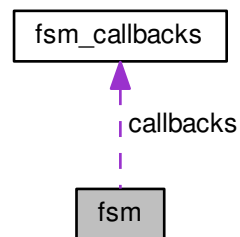
The documentation for this struct was generated from the following file:

- `src/netif/ppp/vj.h`

4.3 fsm Struct Reference

```
#include <fsm.h>
```

Collaboration diagram for fsm:



Data Fields

- int [unit](#)
- u_short [protocol](#)
- int [state](#)
- int [flags](#)
- u_char [id](#)
- u_char [reqid](#)
- u_char [seen_ack](#)
- int [timeouttime](#)
- int [maxconfreqtransmits](#)
- int [retransmits](#)
- int [maxtermtransmits](#)
- int [nakloops](#)
- int [maxnakloops](#)
- struct [fsm_callbacks](#) * [callbacks](#)
- char * [term_reason](#)
- int [term_reason_len](#)

4.3.1 Detailed Description

Definition at line 78 of file fsm.h.

4.3.2 Field Documentation

4.3.2.1 struct [fsm_callbacks](#)* [callbacks](#)

Definition at line 92 of file fsm.h.

4.3.2.2 int [flags](#)

Definition at line 82 of file fsm.h.

4.3.2.3 `u_char id`

Definition at line 83 of file fsm.h.

4.3.2.4 `int maxconfreqtransmits`

Definition at line 87 of file fsm.h.

4.3.2.5 `int maxnakloops`

Definition at line 91 of file fsm.h.

4.3.2.6 `int maxtermtransmits`

Definition at line 89 of file fsm.h.

4.3.2.7 `int nakloops`

Definition at line 90 of file fsm.h.

4.3.2.8 `u_short protocol`

Definition at line 80 of file fsm.h.

4.3.2.9 `u_char reqid`

Definition at line 84 of file fsm.h.

4.3.2.10 `int retransmits`

Definition at line 88 of file fsm.h.

4.3.2.11 `u_char seen_ack`

Definition at line 85 of file fsm.h.

4.3.2.12 `int state`

Definition at line 81 of file fsm.h.

4.3.2.13 `char* term_reason`

Definition at line 93 of file fsm.h.

4.3.2.14 `int term_reason_len`

Definition at line 94 of file fsm.h.

4.3.2.15 int timeouttime

Definition at line 86 of file fsm.h.

4.3.2.16 int unit

Definition at line 79 of file fsm.h.

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

- [src/netif/ppp/fsm.h](#)

4.4 fsm_callbacks Struct Reference

```
#include <fsm.h>
```

Data Fields

- void(* [resetci](#))(fsm *)
- int(* [cilen](#))(fsm *)
- void(* [addci](#))(fsm *, u_char *, int *)
- int(* [ackci](#))(fsm *, u_char *, int)
- int(* [nakci](#))(fsm *, u_char *, int)
- int(* [rejci](#))(fsm *, u_char *, int)
- int(* [reqci](#))(fsm *, u_char *, int *, int)
- void(* [up](#))(fsm *)
- void(* [down](#))(fsm *)
- void(* [starting](#))(fsm *)
- void(* [finished](#))(fsm *)
- void(* [protreject](#))(int)
- void(* [retransmit](#))(fsm *)
- int(* [extcode](#))(fsm *, int, u_char, u_char *, int)
- char * [proto_name](#)

4.4.1 Detailed Description

Definition at line 98 of file fsm.h.

4.4.2 Field Documentation

4.4.2.1 int(* [ackci](#))(fsm *, u_char *, int)

Definition at line 102 of file fsm.h.

4.4.2.2 void(* [addci](#))(fsm *, u_char *, int *)

Definition at line 101 of file fsm.h.

4.4.2.3 int(* [cilen](#))(fsm *)

Definition at line 100 of file fsm.h.

4.4.2.4 void(* down) (fsm *)

Definition at line 107 of file fsm.h.

4.4.2.5 int(* extcode) (fsm *, int, u_char, u_char *, int)

Definition at line 112 of file fsm.h.

4.4.2.6 void(* finished) (fsm *)

Definition at line 109 of file fsm.h.

4.4.2.7 int(* nakci) (fsm *, u_char *, int)

Definition at line 103 of file fsm.h.

4.4.2.8 char* proto_name

Definition at line 113 of file fsm.h.

4.4.2.9 void(* protreject) (int)

Definition at line 110 of file fsm.h.

4.4.2.10 int(* rejci) (fsm *, u_char *, int)

Definition at line 104 of file fsm.h.

4.4.2.11 int(* reqci) (fsm *, u_char *, int *, int)

Definition at line 105 of file fsm.h.

4.4.2.12 void(* resetci) (fsm *)

Definition at line 99 of file fsm.h.

4.4.2.13 void(* retransmit) (fsm *)

Definition at line 111 of file fsm.h.

4.4.2.14 void(* starting) (fsm *)

Definition at line 108 of file fsm.h.

4.4.2.15 void(* up) (fsm *)

Definition at line 106 of file fsm.h.

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

- [src/netif/ppp/fsm.h](#)

4.5 icmp_echo_hdr Struct Reference

```
#include <icmp.h>
```

Public Member Functions

- [PACK_STRUCT_FIELD](#) (u8_t type)
- [PACK_STRUCT_FIELD](#) (u8_t code)
- [PACK_STRUCT_FIELD](#) (u16_t chksum)
- [PACK_STRUCT_FIELD](#) (u16_t id)
- [PACK_STRUCT_FIELD](#) (u16_t seqno)

4.5.1 Detailed Description

This is the standard ICMP header only that the u32_t data is splitted to two u16_t like ICMP echo needs it. This header is also used for other ICMP types that do not use the data part.

Definition at line 79 of file icmp.h.

4.5.2 Member Function Documentation

4.5.2.1 [PACK_STRUCT_FIELD](#) (u8_t *type*)

4.5.2.2 [PACK_STRUCT_FIELD](#) (u8_t *code*)

4.5.2.3 [PACK_STRUCT_FIELD](#) (u16_t *chksum*)

4.5.2.4 [PACK_STRUCT_FIELD](#) (u16_t *id*)

4.5.2.5 [PACK_STRUCT_FIELD](#) (u16_t *seqno*)

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

- src/include/ipv4/lwip/[icmp.h](#)

4.6 in_addr Struct Reference

```
#include <inet.h>
```

Data Fields

- u32_t [s_addr](#)

4.6.1 Detailed Description

For compatibility with BSD code

Definition at line 44 of file inet.h.

4.6.2 Field Documentation

4.6.2.1 u32_t s_addr

Definition at line 45 of file inet.h.

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

- [src/include/ipv4/lwip/inet.h](#)

4.7 ip_addr Struct Reference

```
#include <ip_addr.h>
```

Public Member Functions

- [PACK_STRUCT_FIELD](#) (u32_t addr[4])

Data Fields

- u32_t [addr](#)

4.7.1 Detailed Description

Definition at line 44 of file ip_addr.h.

4.7.2 Member Function Documentation

4.7.2.1 [PACK_STRUCT_FIELD](#) (u32_t addr[4])

4.7.3 Field Documentation

4.7.3.1 u32_t addr

Definition at line 45 of file ip_addr.h.

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

- [src/include/ipv4/lwip/ip_addr.h](#)

4.8 ip_addr2 Struct Reference

```
#include <ip_addr.h>
```

Public Member Functions

- [PACK_STRUCT_FIELD](#) (u16_t addrw[2])
- [PACK_STRUCT_FIELD](#) (u16_t addrw[2])

4.8.1 Detailed Description

Definition at line 75 of file ip_addr.h.

4.8.2 Member Function Documentation

4.8.2.1 `PACK_STRUCT_FIELD (u16_t addrw[2])`

4.8.2.2 `PACK_STRUCT_FIELD (u16_t addrw[2])`

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

- [src/include/ipv4/lwip/ip_addr.h](#)

4.9 ip_addr_packed Struct Reference

```
#include <ip_addr.h>
```

Public Member Functions

- [PACK_STRUCT_FIELD \(u32_t addr\)](#)

4.9.1 Detailed Description

Definition at line 54 of file ip_addr.h.

4.9.2 Member Function Documentation

4.9.2.1 `PACK_STRUCT_FIELD (u32_t addr)`

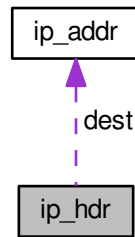
The documentation for this struct was generated from the following file:

- [src/include/ipv4/lwip/ip_addr.h](#)

4.10 ip_hdr Struct Reference

```
#include <ip.h>
```

Collaboration diagram for ip_hdr:



Public Member Functions

- [PACK_STRUCT_FIELD](#) (u8_t _v_hl)
- [PACK_STRUCT_FIELD](#) (u8_t _tos)
- [PACK_STRUCT_FIELD](#) (u16_t _len)
- [PACK_STRUCT_FIELD](#) (u16_t _id)
- [PACK_STRUCT_FIELD](#) (u16_t _offset)
- [PACK_STRUCT_FIELD](#) (u8_t _ttl)
- [PACK_STRUCT_FIELD](#) (u8_t _proto)
- [PACK_STRUCT_FIELD](#) (u16_t _chksum)
- [PACK_STRUCT_FIELD](#) (ip_addr_p_t src)
- [PACK_STRUCT_FIELD](#) (ip_addr_p_t dest)

Data Fields

- u8_t [tclass1](#):4
- u8_t [v](#):4
- u8_t [flow1](#):4
- u8_t [tclass2](#):4
- u16_t [flow2](#)
- u16_t [len](#)
- u8_t [nexthdr](#)
- u8_t [hoplim](#)
- struct [ip_addr](#) src [dest](#)

4.10.1 Detailed Description

Definition at line 116 of file ip.h.

4.10.2 Member Function Documentation

4.10.2.1 [PACK_STRUCT_FIELD](#) (u8_t _v_hl)

4.10.2.2 [PACK_STRUCT_FIELD](#) (u8_t _tos)

4.10.2.3 PACK_STRUCT_FIELD (u16_t len)

4.10.2.4 PACK_STRUCT_FIELD (u16_t id)

4.10.2.5 PACK_STRUCT_FIELD (u16_t offset)

4.10.2.6 PACK_STRUCT_FIELD (u8_t ttl)

4.10.2.7 PACK_STRUCT_FIELD (u8_t proto)

4.10.2.8 PACK_STRUCT_FIELD (u16_t checksum)

4.10.2.9 PACK_STRUCT_FIELD (ip_addr_p_t src)

4.10.2.10 PACK_STRUCT_FIELD (ip_addr_p_t dest)

4.10.3 Field Documentation

4.10.3.1 struct ip_addr src dest

Definition at line 96 of file ip.h.

4.10.3.2 u8_t flow1

Definition at line 87 of file ip.h.

4.10.3.3 u16_t flow2

Definition at line 92 of file ip.h.

4.10.3.4 u8_t hoplim

Definition at line 95 of file ip.h.

4.10.3.5 u16_t len

Definition at line 93 of file ip.h.

4.10.3.6 u8_t nexthdr

Definition at line 94 of file ip.h.

4.10.3.7 u8_t tclass1

Definition at line 86 of file ip.h.

4.10.3.8 u8_t tclass2

Definition at line 87 of file ip.h.

4.10.3.9 u8_t v

Definition at line 86 of file ip.h.

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

- [src/include/ipv4/lwip/ip.h](#)

4.11 ip_pcb Struct Reference

```
#include <ip.h>
```

Data Fields

- [IP_PCB](#)

4.11.1 Detailed Description

Definition at line 89 of file ip.h.

4.11.2 Field Documentation

4.11.2.1 IP_PCB

Definition at line 91 of file ip.h.

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

- [src/include/ipv4/lwip/ip.h](#)

4.12 ipcp_options Struct Reference

```
#include <ipcp.h>
```

Data Fields

- u_int [neg_addr](#): 1
- u_int [old_addrs](#): 1
- u_int [req_addr](#): 1
- u_int [default_route](#): 1
- u_int [proxy_arp](#): 1
- u_int [neg_vj](#): 1
- u_int [old_vj](#): 1
- u_int [accept_local](#): 1
- u_int [accept_remote](#): 1
- u_int [req_dns1](#): 1
- u_int [req_dns2](#): 1
- u_short [vj_protocol](#)
- u_char [maxslotindex](#)
- u_char [cflag](#)
- u32_t [ouraddr](#)

- u32_t [hisaddr](#)
- u32_t [dnsaddr](#) [2]
- u32_t [winsaddr](#) [2]

4.12.1 Detailed Description

Definition at line 78 of file ipcp.h.

4.12.2 Field Documentation

4.12.2.1 u_int accept_local

Definition at line 86 of file ipcp.h.

4.12.2.2 u_int accept_remote

Definition at line 87 of file ipcp.h.

4.12.2.3 u_char cflag

Definition at line 92 of file ipcp.h.

4.12.2.4 u_int default_route

Definition at line 82 of file ipcp.h.

4.12.2.5 u32_t dnsaddr[2]

Definition at line 94 of file ipcp.h.

4.12.2.6 u32_t hisaddr

Definition at line 93 of file ipcp.h.

4.12.2.7 u_char maxslotindex

Definition at line 91 of file ipcp.h.

4.12.2.8 u_int neg_addr

Definition at line 79 of file ipcp.h.

4.12.2.9 u_int neg_vj

Definition at line 84 of file ipcp.h.

4.12.2.10 u_int old_addrs

Definition at line 80 of file ipcp.h.

4.12.2.11 u_int old_vj

Definition at line 85 of file ipcp.h.

4.12.2.12 u32_t ouraddr

Definition at line 93 of file ipcp.h.

4.12.2.13 u_int proxy_arp

Definition at line 83 of file ipcp.h.

4.12.2.14 u_int req_addr

Definition at line 81 of file ipcp.h.

4.12.2.15 u_int req_dns1

Definition at line 88 of file ipcp.h.

4.12.2.16 u_int req_dns2

Definition at line 89 of file ipcp.h.

4.12.2.17 u_short vj_protocol

Definition at line 90 of file ipcp.h.

4.12.2.18 u32_t winsaddr[2]

Definition at line 95 of file ipcp.h.

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

- [src/netif/ppp/ipcp.h](#)

4.13 lcp_options Struct Reference

```
#include <lcp.h>
```

Data Fields

- u_int [passive](#): 1
- u_int [silent](#): 1
- u_int [restart](#): 1
- u_int [neg_mru](#): 1
- u_int [neg_asyncmap](#): 1
- u_int [neg_upap](#): 1
- u_int [neg_chap](#): 1
- u_int [neg_magicnumber](#): 1

- u_int [neg_pcompression](#): 1
- u_int [neg_accompression](#): 1
- u_int [neg_lqr](#): 1
- u_int [neg_cbc](#): 1
- u_short [mru](#)
- u_char [chap_mdtype](#)
- u32_t [asynmap](#)
- u32_t [magicnumber](#)
- int [numloops](#)
- u32_t [lqr_period](#)

4.13.1 Detailed Description

Definition at line 83 of file lcp.h.

4.13.2 Field Documentation

4.13.2.1 u32_t asynmap

Definition at line 106 of file lcp.h.

4.13.2.2 u_char chap_mdtype

Definition at line 105 of file lcp.h.

4.13.2.3 u32_t lqr_period

Definition at line 109 of file lcp.h.

4.13.2.4 u32_t magicnumber

Definition at line 107 of file lcp.h.

4.13.2.5 u_short mru

Definition at line 101 of file lcp.h.

4.13.2.6 u_int neg_accompression

Definition at line 93 of file lcp.h.

4.13.2.7 u_int neg_asynmap

Definition at line 88 of file lcp.h.

4.13.2.8 u_int neg_cbc

Definition at line 95 of file lcp.h.

4.13.2.9 u_int neg_chap

Definition at line 90 of file lcp.h.

4.13.2.10 u_int neg_lqr

Definition at line 94 of file lcp.h.

4.13.2.11 u_int neg_magicnumber

Definition at line 91 of file lcp.h.

4.13.2.12 u_int neg_mru

Definition at line 87 of file lcp.h.

4.13.2.13 u_int neg_pcompression

Definition at line 92 of file lcp.h.

4.13.2.14 u_int neg_upap

Definition at line 89 of file lcp.h.

4.13.2.15 int numloops

Definition at line 108 of file lcp.h.

4.13.2.16 u_int passive

Definition at line 84 of file lcp.h.

4.13.2.17 u_int restart

Definition at line 86 of file lcp.h.

4.13.2.18 u_int silent

Definition at line 85 of file lcp.h.

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

- [src/netif/ppp/lcp.h](#)

4.14 MD5_CTX Struct Reference

```
#include <md5.h>
```


Data Fields

- `u32_t i` [2]
- `u32_t buf` [4]
- unsigned char `in` [64]
- unsigned char `digest` [16]

4.14.1 Detailed Description

Definition at line 44 of file `md5.h`.

4.14.2 Field Documentation

4.14.2.1 `u32_t buf[4]`

Definition at line 46 of file `md5.h`.

4.14.2.2 unsigned char `digest[16]`

Definition at line 48 of file `md5.h`.

4.14.2.3 `u32_t i[2]`

Definition at line 45 of file `md5.h`.

4.14.2.4 unsigned char `in[64]`

Definition at line 47 of file `md5.h`.

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

- `src/netif/ppp/md5.h`

4.15 mem Struct Reference

Data Fields

- `mem_size_t next`
- `mem_size_t prev`
- `u8_t used`

4.15.1 Detailed Description

The heap is made up as a list of structs of this type. This does not have to be aligned since for getting its size, we only use the macro `SIZEOF_STRUCT_MEM`, which automatically alignes.

Definition at line 156 of file `mem.c`.

4.15.2 Field Documentation

4.15.2.1 `mem_size_t` next

index (-> ram[next]) of the next struct

Definition at line 158 of file mem.c.

4.15.2.2 `mem_size_t` prev

index (-> ram[prev]) of the previous struct

Definition at line 160 of file mem.c.

4.15.2.3 `u8_t` used

1: this area is used; 0: this area is unused

Definition at line 162 of file mem.c.

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

- src/core/[mem.c](#)

4.16 memp Struct Reference

Collaboration diagram for memp:



Data Fields

- struct [memp](#) * [next](#)

4.16.1 Detailed Description

Definition at line 66 of file memp.c.

4.16.2 Field Documentation

4.16.2.1 `struct memp*` next

Definition at line 67 of file memp.c.

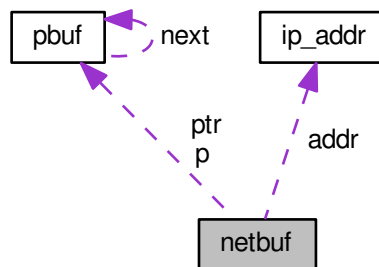
The documentation for this struct was generated from the following file:

- src/core/[memp.c](#)

4.17 netbuf Struct Reference

```
#include <netbuf.h>
```

Collaboration diagram for netbuf:



Data Fields

- struct `pbuf` * `p`
- struct `pbuf` * `ptr`
- `ip_addr_t` `addr`
- `u16_t` `port`

4.17.1 Detailed Description

Definition at line 48 of file `netbuf.h`.

4.17.2 Field Documentation

4.17.2.1 `ip_addr_t` `addr`

Definition at line 50 of file `netbuf.h`.

4.17.2.2 struct `pbuf`* `p`

Definition at line 49 of file `netbuf.h`.

4.17.2.3 `u16_t` `port`

Definition at line 51 of file `netbuf.h`.

4.17.2.4 struct `pbuf` * `ptr`

Definition at line 49 of file `netbuf.h`.

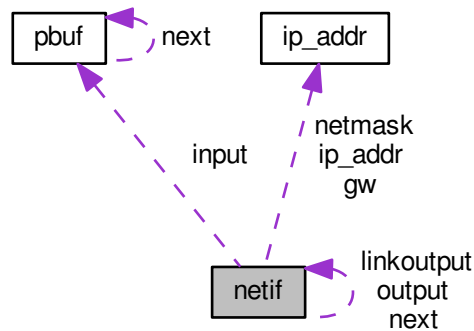
The documentation for this struct was generated from the following file:

- `src/include/lwip/netbuf.h`

4.18 netif Struct Reference

```
#include <netif.h>
```

Collaboration diagram for netif:



Data Fields

- struct `netif` * `next`
- `ip_addr_t` `ip_addr`
- `ip_addr_t` `netmask`
- `ip_addr_t` `gw`
- `netif_input_fn` `input`
- `netif_output_fn` `output`
- `netif_linkoutput_fn` `linkoutput`
- void * `state`
- `u16_t` `mtu`
- `u8_t` `hwaddr_len`
- `u8_t` `hwaddr` [`NETIF_MAX_HWADDR_LEN`]
- `u8_t` `flags`
- char `name` [2]
- `u8_t` `num`

4.18.1 Detailed Description

Generic data structure used for all lwIP network interfaces. The following fields should be filled in by the initialization function for the device driver: `hwaddr_len`, `hwaddr[]`, `mtu`, `flags`

Definition at line 136 of file `netif.h`.

4.18.2 Field Documentation

4.18.2.1 `u8_t` `flags`

`flags` (see `NETIF_FLAG_` above)

Definition at line 192 of file `netif.h`.

4.18.2.2 `ip_addr_t gw`

Definition at line 143 of file netif.h.

4.18.2.3 `u8_t hwaddr[NETIF_MAX_HWADDR_LEN]`

link level hardware address of this interface

Definition at line 190 of file netif.h.

4.18.2.4 `u8_t hwaddr_len`

number of bytes used in hwaddr

Definition at line 188 of file netif.h.

4.18.2.5 `netif_input_fn input`

This function is called by the network device driver to pass a packet up the TCP/IP stack.

Definition at line 147 of file netif.h.

4.18.2.6 `ip_addr_t ip_addr`

IP address configuration in network byte order

Definition at line 141 of file netif.h.

4.18.2.7 `netif_linkoutput_fn linkoutput`

This function is called by the ARP module when it wants to send a packet on the interface. This function outputs the pbuf as-is on the link medium.

Definition at line 155 of file netif.h.

4.18.2.8 `u16_t mtu`

maximum transfer unit (in bytes)

Definition at line 186 of file netif.h.

4.18.2.9 `char name[2]`

descriptive abbreviation

Definition at line 194 of file netif.h.

4.18.2.10 `ip_addr_t netmask`

Definition at line 142 of file netif.h.

4.18.2.11 `struct netif* next`

pointer to next in linked list

Definition at line 138 of file netif.h.

4.18.2.12 `u8_t num`

number of this interface

Definition at line 196 of file `netif.h`.

4.18.2.13 `netif_output_fn` output

This function is called by the IP module when it wants to send a packet on the interface. This function typically first resolves the hardware address, then sends the packet.

Definition at line 151 of file `netif.h`.

4.18.2.14 `void* state`

This field can be set by the device driver and could point to state information for the device.

Definition at line 172 of file `netif.h`.

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

- `src/include/lwip/netif.h`

4.19 `pbuf` Struct Reference

```
#include <pbuf.h>
```

Collaboration diagram for `pbuf`:



Data Fields

- struct `pbuf` * `next`
- void * `payload`
- `u16_t` `tot_len`
- `u16_t` `len`
- `u8_t` `type`
- `u8_t` `flags`
- `u16_t` `ref`

4.19.1 Detailed Description

Definition at line 79 of file `pbuf.h`.

4.19.2 Field Documentation

4.19.2.1 u8_t flags

misc flags

Definition at line 102 of file pbuf.h.

4.19.2.2 u16_t len

length of this buffer

Definition at line 96 of file pbuf.h.

4.19.2.3 struct pbuf* next

next pbuf in singly linked pbuf chain

Definition at line 81 of file pbuf.h.

4.19.2.4 void* payload

pointer to the actual data in the buffer

Definition at line 84 of file pbuf.h.

4.19.2.5 u16_t ref

the reference count always equals the number of pointers that refer to this pbuf. This can be pointers from an application, the stack itself, or pbuf->next pointers from a chain.

Definition at line 109 of file pbuf.h.

4.19.2.6 u16_t tot_len

total length of this buffer and all next buffers in chain belonging to the same packet.

For non-queue packet chains this is the invariant: p->tot_len == p->len + (p->next? p->next->tot_len: 0)

Definition at line 93 of file pbuf.h.

4.19.2.7 u8_t type

pbuf_type as u8_t instead of enum to save space

Definition at line 99 of file pbuf.h.

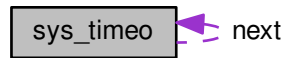
The documentation for this struct was generated from the following file:

- src/include/lwip/pbuf.h

4.20 sys_timeo Struct Reference

```
#include <timers.h>
```

Collaboration diagram for sys_timeo:



Data Fields

- struct [sys_timeo](#) * [next](#)
- [u32_t](#) [time](#)
- [sys_timeout_handler](#) [h](#)
- void * [arg](#)

4.20.1 Detailed Description

Definition at line 67 of file [timers.h](#).

4.20.2 Field Documentation

4.20.2.1 void* arg

Definition at line 71 of file [timers.h](#).

4.20.2.2 sys_timeout_handler h

Definition at line 70 of file [timers.h](#).

4.20.2.3 struct sys_timeo* next

Definition at line 68 of file [timers.h](#).

4.20.2.4 u32_t time

Definition at line 69 of file [timers.h](#).

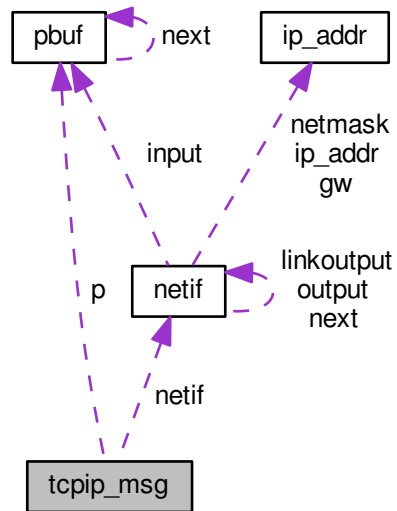
The documentation for this struct was generated from the following file:

- [src/include/lwip/timers.h](#)

4.21 tcpip_msg Struct Reference

```
#include <tcpip.h>
```


Collaboration diagram for tcpip_msg:



Data Fields

- enum `tcpip_msg_type` `type`
- `sys_sem_t` * `sem`
- union {
 - struct {
 - struct `pbuf` * `p`
 - struct `netif` * `netif`
 - `inp`
 - struct {
 - `tcpip_callback_fn` function
 - void * `ctx`
 - `cb`
- } `msg`

4.21.1 Detailed Description

Definition at line 133 of file `tcpip.h`.

4.21.2 Field Documentation

4.21.2.1 struct { ... } `cb`

4.21.2.2 void* `ctx`

Definition at line 149 of file `tcpip.h`.

4.21.2.3 `tcpip_callback_fn` function

Definition at line 148 of file `tcpip.h`.

4.21.2.4 `struct { ... } inp`

4.21.2.5 `union { ... } msg`

4.21.2.6 `struct netif* netif`

Definition at line 145 of file `tcpip.h`.

4.21.2.7 `struct pbuf* p`

Definition at line 144 of file `tcpip.h`.

4.21.2.8 `sys_sem_t* sem`

Definition at line 135 of file `tcpip.h`.

4.21.2.9 `enum tcpip_msg_type` type

Definition at line 134 of file `tcpip.h`.

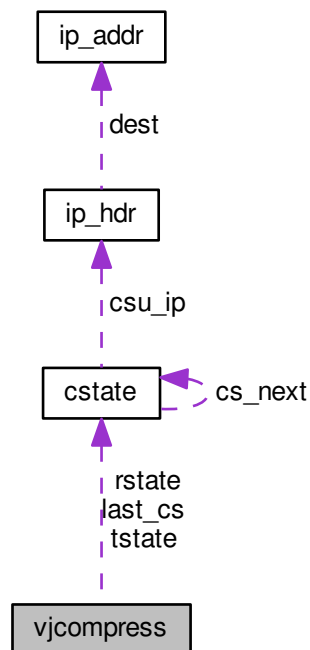
The documentation for this struct was generated from the following file:

- `src/include/lwip/tcpip.h`

4.22 `vjcompress` Struct Reference

```
#include <vj.h>
```

Collaboration diagram for vjcompress:



Data Fields

- struct `cstate` * `last_cs`
- u_char `last_rcv`
- u_char `last_xmit`
- u_short `flags`
- u_char `maxSlotIndex`
- u_char `compressSlot`
- struct `cstate` `tstate` [`MAX_SLOTS`]
- struct `cstate` `rstate` [`MAX_SLOTS`]

4.22.1 Detailed Description

Definition at line 133 of file `vj.h`.

4.22.2 Field Documentation

4.22.2.1 u_char compressSlot

Definition at line 139 of file `vj.h`.

4.22.2.2 u_short flags

Definition at line 137 of file `vj.h`.

4.22.2.3 struct cstate* last_cs

Definition at line 134 of file vj.h.

4.22.2.4 u_char last_recv

Definition at line 135 of file vj.h.

4.22.2.5 u_char last_xmit

Definition at line 136 of file vj.h.

4.22.2.6 u_char maxSlotIndex

Definition at line 138 of file vj.h.

4.22.2.7 struct cstate rstate[MAX_SLOTS]

Definition at line 144 of file vj.h.

4.22.2.8 struct cstate tstate[MAX_SLOTS]

Definition at line 143 of file vj.h.

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

- [src/netif/ppp/vj.h](#)

4.23 vjstat Struct Reference

```
#include <vj.h>
```

Data Fields

- unsigned long [vjs_packets](#)
- unsigned long [vjs_compressed](#)
- unsigned long [vjs_searches](#)
- unsigned long [vjs_misses](#)
- unsigned long [vjs_uncompressedin](#)
- unsigned long [vjs_compressedin](#)
- unsigned long [vjs_errorin](#)
- unsigned long [vjs_tossed](#)

4.23.1 Detailed Description

Definition at line 119 of file vj.h.

4.23.2 Field Documentation

4.23.2.1 unsigned long vjs_compressed

Definition at line 121 of file vj.h.

4.23.2.2 unsigned long vjs_compressedin

Definition at line 125 of file vj.h.

4.23.2.3 unsigned long vjs_errorin

Definition at line 126 of file vj.h.

4.23.2.4 unsigned long vjs_misses

Definition at line 123 of file vj.h.

4.23.2.5 unsigned long vjs_packets

Definition at line 120 of file vj.h.

4.23.2.6 unsigned long vjs_searches

Definition at line 122 of file vj.h.

4.23.2.7 unsigned long vjs_tossed

Definition at line 127 of file vj.h.

4.23.2.8 unsigned long vjs_uncompressedin

Definition at line 124 of file vj.h.

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

- [src/netif/ppp/vj.h](#)

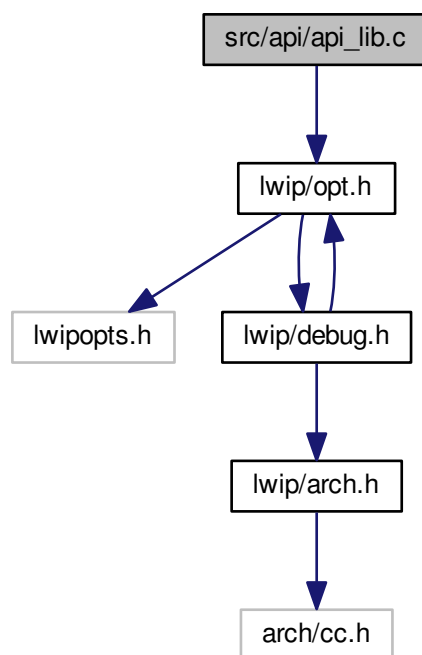
Chapter 5

File Documentation

5.1 src/api/api_lib.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for api_lib.c:



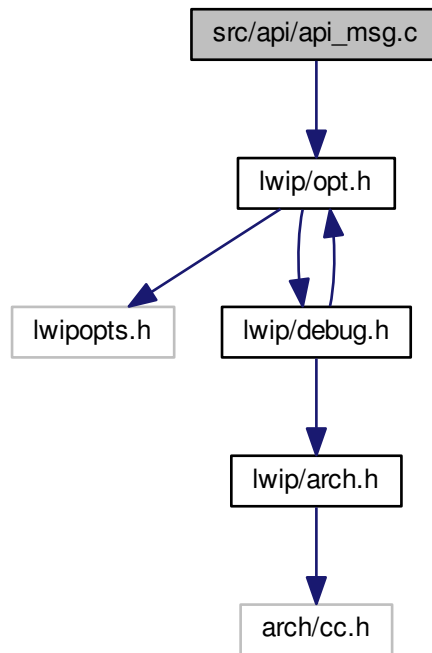
5.1.1 Detailed Description

Sequential API External module

5.2 src/api/api_msg.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for api_msg.c:



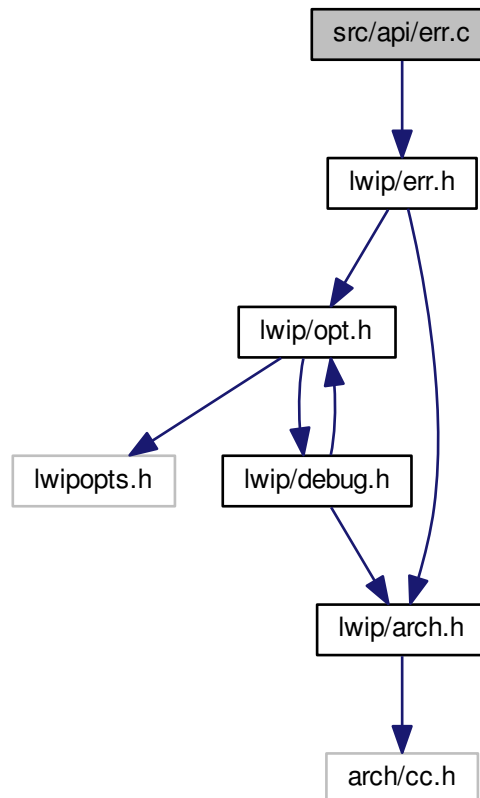
5.2.1 Detailed Description

Sequential API Internal module

5.3 src/api/err.c File Reference

```
#include "lwip/err.h"
```

Include dependency graph for err.c:



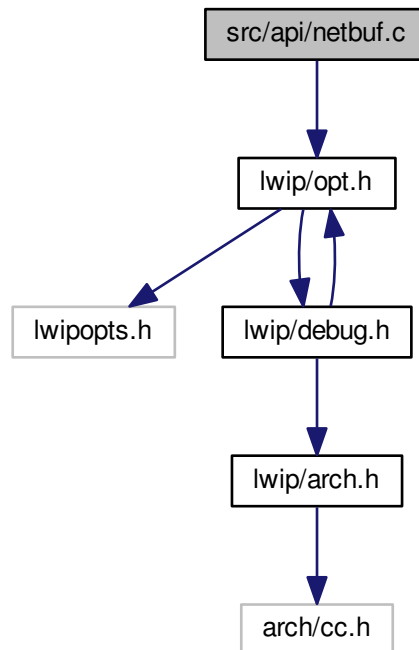
5.3.1 Detailed Description

Error Management module

5.4 src/api/netbuf.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for netbuf.c:

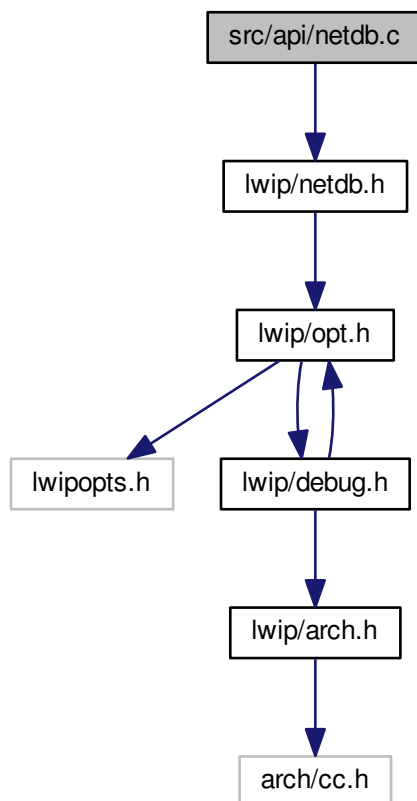


5.4.1 Detailed Description

Network buffer management

5.5 src/api/netdb.c File Reference

```
#include "lwip/netdb.h"
Include dependency graph for netdb.c:
```



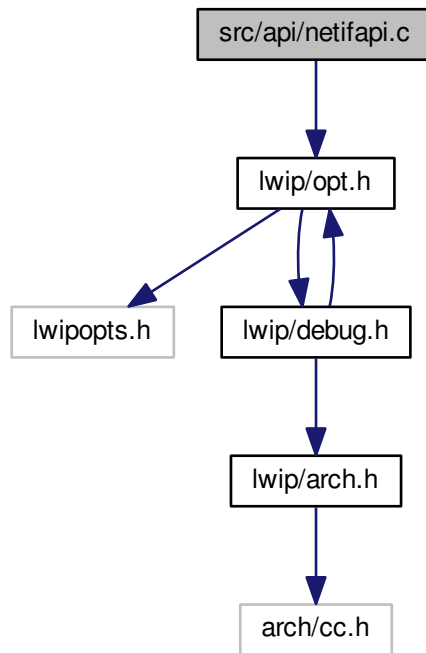
5.5.1 Detailed Description

API functions for name resolving

5.6 src/api/netifapi.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for netifapi.c:



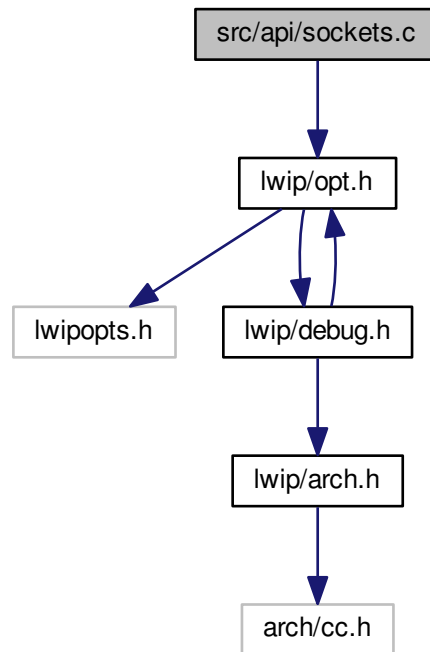
5.6.1 Detailed Description

Network Interface Sequential API module

5.7 src/api/sockets.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for sockets.c:



5.7.1 Detailed Description

Sockets BSD-Like API module

5.8 src/api/tcpip.c File Reference

```
#include "lwip/opt.h"
#include "lwip/sys.h"
#include "lwip/memp.h"
#include "lwip/mem.h"
#include "lwip/pbuf.h"
#include "lwip/tcpip.h"
#include "lwip/init.h"
#include "netif/etharp.h"
#include "netif/ppp_oe.h"
```


Parameters

<i>p</i>	The pbuf (chain) to be dereferenced.
----------	--------------------------------------

Returns

ERR_OK if callback could be enqueued, an err_t if not

Definition at line 493 of file tcpip.c.

5.8.2.3 err_t tcpip_callback_with_block (tcpip_callback_fn function, void * ctx, u8_t block)

Call a specific function in the thread context of tcpip_thread for easy access synchronization. A function called in that way may access lwIP core code without fearing concurrent access.

Parameters

<i>f</i>	the function to call
<i>ctx</i>	parameter passed to f
<i>block</i>	1 to block until the request is posted, 0 to non-blocking mode

Returns

ERR_OK if the function was called, another err_t if not

Definition at line 211 of file tcpip.c.

5.8.2.4 void tcpip_callbackmsg_delete (struct tcpip_callback_msg * msg)

Free a callback message allocated by [tcpip_callbackmsg_new\(\)](#).

Parameters

<i>msg</i>	the message to free
------------	---------------------

Definition at line 425 of file tcpip.c.

5.8.2.5 struct tcpip_callback_msg* tcpip_callbackmsg_new (tcpip_callback_fn function, void * ctx)

Allocate a structure for a static callback message and initialize it. This is intended to be used to send "static" messages from interrupt context.

Parameters

<i>function</i>	the function to call
<i>ctx</i>	parameter passed to function

Returns

a struct pointer to pass to [tcpip_trycallback\(\)](#).

Definition at line 408 of file tcpip.c.

5.8.2.6 void tcpip_init (tcpip_init_done_fn initfunc, void * arg)

Initialize this module:

- initialize all sub modules
- start the tcpip_thread

Parameters

<i>initfunc</i>	a function to call when tcpip_thread is running and finished initializing
<i>arg</i>	argument to pass to initfunc

Definition at line 455 of file tcpip.c.

5.8.2.7 err_t tcpip_input (struct pbuf * p, struct netif * inp)

Pass a received packet to tcpip_thread for input processing

Parameters

<i>p</i>	the received packet, p->payload pointing to the Ethernet header or to an IP header (if inp doesn't have NETIF_FLAG_ETHARP or NETIF_FLAG_ETHERNET flags)
<i>inp</i>	the network interface on which the packet was received

Definition at line 161 of file tcpip.c.

5.8.2.8 err_t tcpip_trycallback (struct tcpip_callback_msg * msg)

Try to post a callback-message to the tcpip_thread mbox This is intended to be used to send "static" messages from interrupt context.

Parameters

<i>msg</i>	pointer to the message to post
------------	--------------------------------

Returns

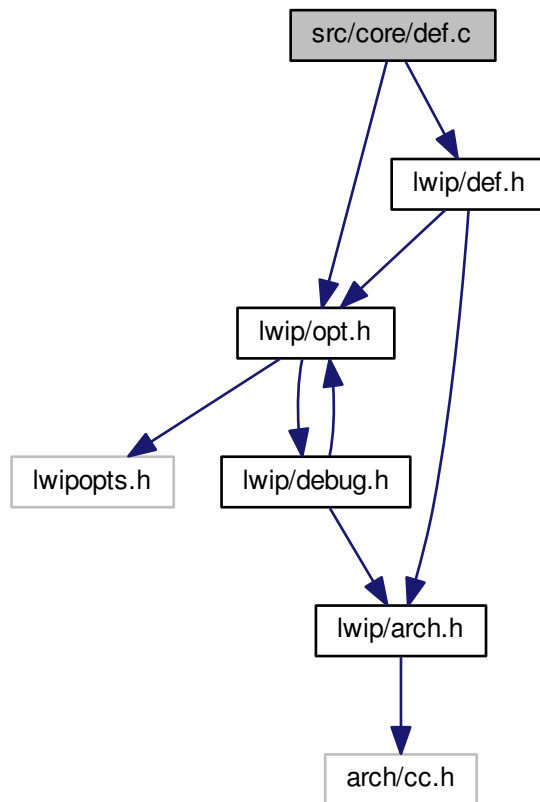
[sys_mbox_trypost\(\)](#) return code

Definition at line 438 of file tcpip.c.

5.9 src/core/def.c File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
```


Include dependency graph for def.c:



Functions

- `u16_t` [lwip_htons](#) (`u16_t n`)
- `u16_t` [lwip_ntohs](#) (`u16_t n`)
- `u32_t` [lwip_htonl](#) (`u32_t n`)
- `u32_t` [lwip_ntohl](#) (`u32_t n`)

5.9.1 Detailed Description

Common functions used throughout the stack.

5.9.2 Function Documentation

5.9.2.1 `u32_t` `lwip_htonl` (`u32_t n`)

Convert an `u32_t` from host- to network byte order.

Parameters

<i>n</i>	u32_t in host byte order
----------	--------------------------

Returns

n in network byte order

Definition at line 88 of file def.c.

5.9.2.2 u16_t lwip_htons (u16_t *n*)

These are reference implementations of the byte swapping functions. Again with the aim of being simple, correct and fully portable. Byte swapping is the second thing you would want to optimize. You will need to port it to your architecture and in your cc.h:

```
#define LWIP_PLATFORM_BYTESWAP 1 #define LWIP_PLATFORM_HTONS(x) <your_htons> #define LWIP_PLATFORM_HTONL(x) <your_htonl>
```

Note [ntohs\(\)](#) and [ntohl\(\)](#) are merely references to the htonx counterparts. Convert an u16_t from host- to network byte order.

Parameters

<i>n</i>	u16_t in host byte order
----------	--------------------------

Returns

n in network byte order

Definition at line 64 of file def.c.

5.9.2.3 u32_t lwip_ntohl (u32_t *n*)

Convert an u32_t from network- to host byte order.

Parameters

<i>n</i>	u32_t in network byte order
----------	-----------------------------

Returns

n in host byte order

Definition at line 103 of file def.c.

5.9.2.4 u16_t lwip_ntohs (u16_t *n*)

Convert an u16_t from network- to host byte order.

Parameters

<i>n</i>	u16_t in network byte order
----------	-----------------------------

Returns

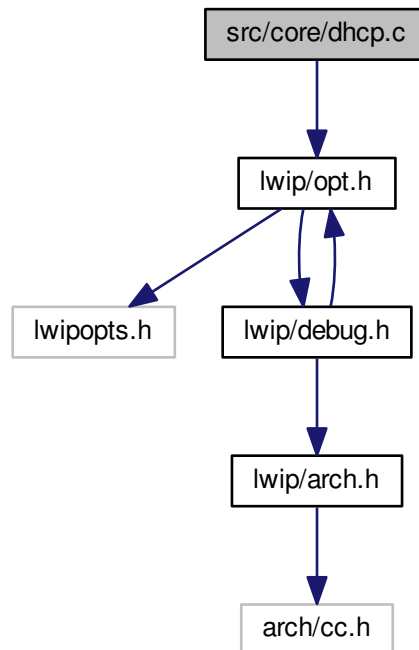
n in host byte order

Definition at line 76 of file def.c.

5.10 src/core/dhcp.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for dhcp.c:



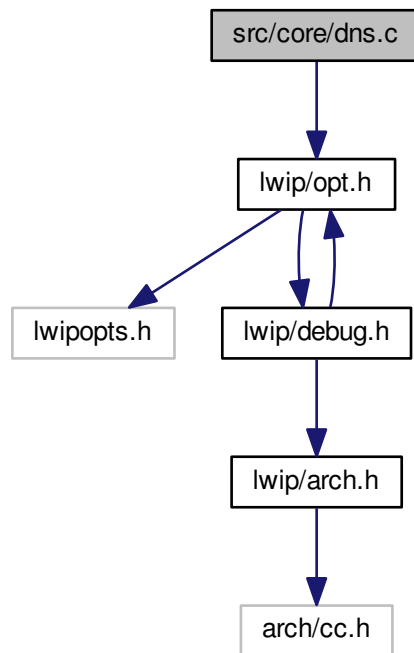
5.10.1 Detailed Description

Dynamic Host Configuration Protocol client

5.11 src/core/dns.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for dns.c:



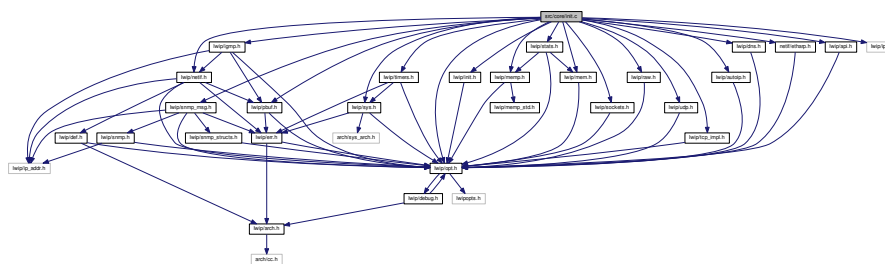
5.11.1 Detailed Description

DNS - host name to IP address resolver.

5.12 src/core/init.c File Reference

```
#include "lwip/opt.h"
#include "lwip/init.h"
#include "lwip/stats.h"
#include "lwip/sys.h"
#include "lwip/mem.h"
#include "lwip/memp.h"
#include "lwip/pbuf.h"
#include "lwip/netif.h"
#include "lwip/sockets.h"
#include "lwip/ip.h"
#include "lwip/raw.h"
#include "lwip/udp.h"
#include "lwip/tcp_impl.h"
#include "lwip/snmp_msg.h"
#include "lwip/autoip.h"
#include "lwip/igmp.h"
#include "lwip/dns.h"
#include "lwip/timers.h"
#include "netif/etharp.h"
#include "lwip/api.h"
```

Include dependency graph for init.c:



Macros

- `#define LWIP_DISABLE_TCP_SANITY_CHECKS 0`
- `#define LWIP_DISABLE_MEMP_SANITY_CHECKS 0`

Functions

- void `lwip_init` (void)

5.12.1 Detailed Description

Modules initialization

5.12.2 Macro Definition Documentation

5.12.2.1 `#define LWIP_DISABLE_MEMP_SANITY_CHECKS 0`

Definition at line 240 of file `init.c`.

5.12.2.2 `#define LWIP_DISABLE_TCP_SANITY_CHECKS 0`

Definition at line 237 of file `init.c`.

5.12.3 Function Documentation

5.12.3.1 `void lwip_init (void)`

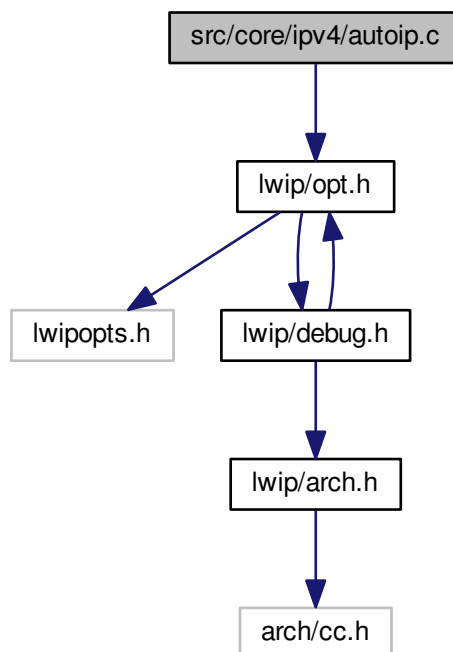
Perform Sanity check of user-configurable values, and initialize all modules.

Definition at line 289 of file `init.c`.

5.13 `src/core/ipv4/autoip.c` File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for `autoip.c`:



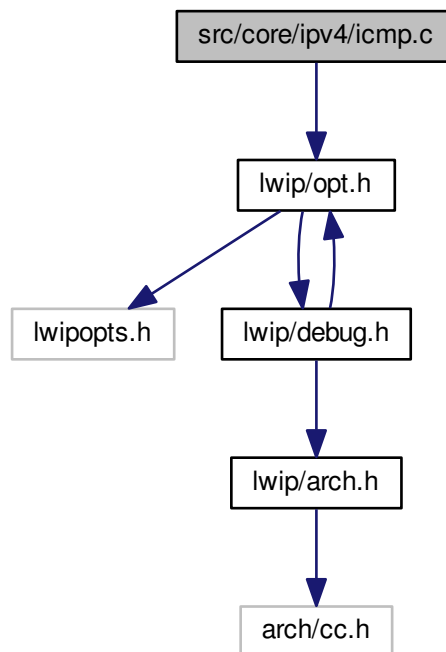
5.13.1 Detailed Description

AutoIP Automatic LinkLocal IP Configuration

5.14 src/core/ipv4/icmp.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for icmp.c:



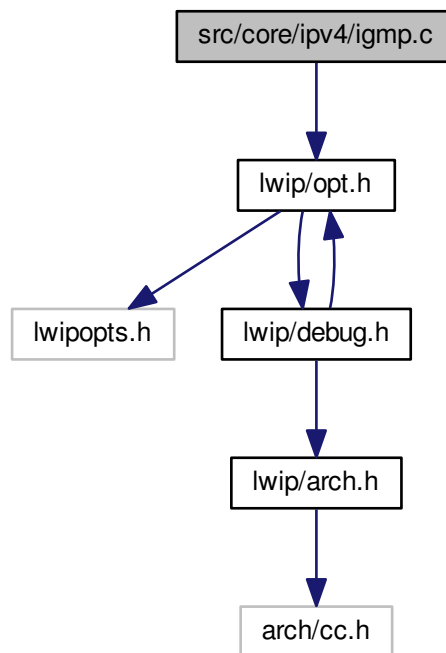
5.14.1 Detailed Description

ICMP - Internet Control Message Protocol

5.15 src/core/ipv4/igmp.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for igmp.c:



5.15.1 Detailed Description

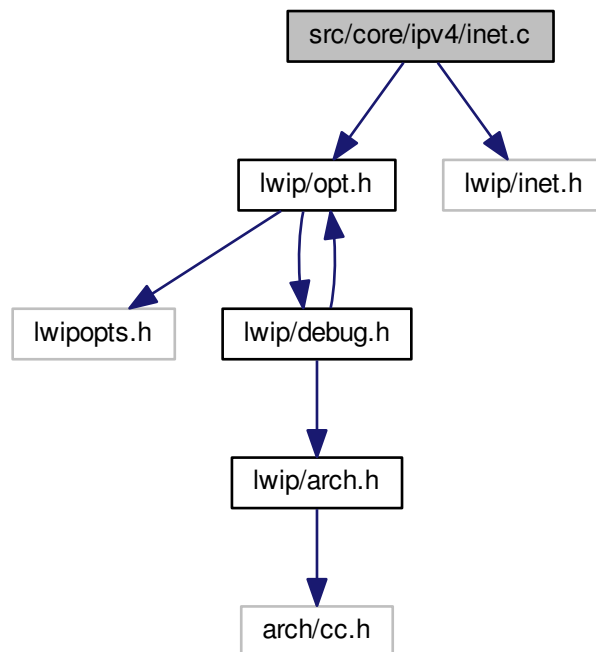
IGMP - Internet Group Management Protocol

5.16 src/core/ipv4/inet.c File Reference

```
#include "lwip/opt.h"
```

```
#include "lwip/inet.h"
```


Include dependency graph for inet.c:



5.16.1 Detailed Description

Functions common to all TCP/IPv4 modules, such as the byte order functions.

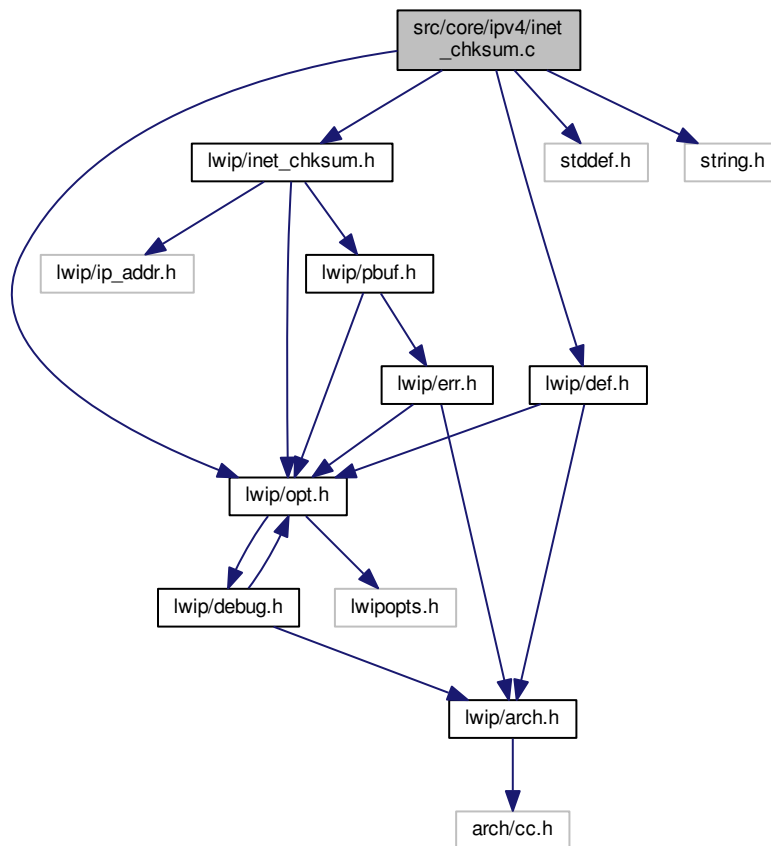
5.17 src/core/ipv4/inet_chksum.c File Reference

```

#include "lwip/opt.h"
#include "lwip/inet_chksum.h"
#include "lwip/def.h"
#include <stddef.h>
#include <string.h>

```

Include dependency graph for inet_chksum.c:



Macros

- `#define LWIP_CHKSUM lwip_standard_chksum`
- `#define LWIP_CHKSUM_ALGORITHM 2`

Functions

- `u16_t inet_chksum_pseudo (struct pbuf *p, ip_addr_t *src, ip_addr_t *dest, u8_t proto, u16_t proto_len)`
- `u16_t inet_chksum_pseudo_partial (struct pbuf *p, ip_addr_t *src, ip_addr_t *dest, u8_t proto, u16_t proto_len, u16_t checksum_len)`
- `u16_t inet_chksum (void *dataptr, u16_t len)`
- `u16_t inet_chksum_pbuf (struct pbuf *p)`

5.17.1 Detailed Description

Include internet checksum functions.

5.17.2 Macro Definition Documentation

5.17.2.1 `#define LWIP_CHKSUM lwip_standard_chksum`

Definition at line 59 of file `inet_chksum.c`.

5.17.2.2 `#define LWIP_CHKSUM_ALGORITHM 2`

Definition at line 61 of file `inet_chksum.c`.

5.17.3 Function Documentation

5.17.3.1 `u16_t inet_chksum (void * dataptr, u16_t len)`

Definition at line 396 of file `inet_chksum.c`.

5.17.3.2 `u16_t inet_chksum_pbuf (struct pbuf * p)`

Calculate a checksum over a chain of pbufs (without pseudo-header, much like `inet_chksum` only pbufs are used).

Parameters

<code>p</code>	pbuf chain over that the checksum should be calculated
----------------	--

Returns

checksum (as `u16_t`) to be saved directly in the protocol header

Definition at line 409 of file `inet_chksum.c`.

5.17.3.3 `u16_t inet_chksum_pseudo (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t proto, u16_t proto_len)`

Definition at line 272 of file `inet_chksum.c`.

5.17.3.4 `u16_t inet_chksum_pseudo_partial (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t proto, u16_t proto_len, u16_t checksum_len)`

Definition at line 332 of file `inet_chksum.c`.

5.18.1 Detailed Description

This is the IPv4 layer implementation for incoming and outgoing IP traffic.

See also

[ip_frag.c](#)

5.18.2 Macro Definition Documentation

5.18.2.1 `#define CHECKSUM_GEN_IP_INLINE 0`

Definition at line 69 of file ip.c.

5.18.2.2 `#define IP_ACCEPT_LINK_LAYER_ADDRESSING 0`

Definition at line 93 of file ip.c.

5.18.2.3 `#define LWIP_INLINE_IP_CHKSUM 1`

Set this to 0 in the rare case of wanting to call an extra function to generate the IP checksum (in contrast to calculating it on-the-fly).

Definition at line 64 of file ip.c.

5.18.3 Function Documentation

5.18.3.1 `err_t ip_input (struct pbuf * p, struct netif * inp)`

This function is called by the network interface device driver when an IP packet is received. The function does the basic checks of the IP header such as packet size being at least larger than the header size etc. If the packet was not destined for us, the packet is forwarded (using `ip_forward`). The IP checksum is always checked.

Finally, the packet is sent to the upper layer protocol input function.

Parameters

<i>p</i>	the received IP packet (p->payload points to IP header)
<i>inp</i>	the netif on which this packet was received

Returns

ERR_OK if the packet was processed (could return ERR_* if it wasn't processed, but currently always returns ERR_OK)

Definition at line 305 of file ip.c.

5.18.3.2 `err_t ip_output (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t ttl, u8_t tos, u8_t proto)`

Simple interface to `ip_output_if`. It finds the outgoing network interface and calls upon `ip_output_if` to do the actual work.

Parameters

<i>p</i>	the packet to send (p->payload points to the data, e.g. next protocol header; if dest == IP_HDRINCL, p already includes an IP header and p->payload points to that IP header)
<i>src</i>	the source IP address to send from (if src == IP_ADDR_ANY, the IP address of the netif used to send is used as source address)
<i>dest</i>	the destination IP address to send the packet to
<i>ttl</i>	the TTL value to be set in the IP header
<i>tos</i>	the TOS value to be set in the IP header
<i>proto</i>	the PROTOCOL to be set in the IP header

Returns

ERR_RTE if no route is found see [ip_output_if\(\)](#) for more return values

Definition at line 818 of file ip.c.

5.18.3.3 `err_t ip_output_if (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t ttl, u8_t tos, u8_t proto, struct netif * netif)`

Sends an IP packet on a network interface. This function constructs the IP header and calculates the IP header checksum. If the source IP address is NULL, the IP address of the outgoing network interface is filled in as source address. If the destination IP address is IP_HDRINCL, p is assumed to already include an IP header and p->payload points to it instead of the data.

Parameters

<i>p</i>	the packet to send (p->payload points to the data, e.g. next protocol header; if dest == IP_HDRINCL, p already includes an IP header and p->payload points to that IP header)
<i>src</i>	the source IP address to send from (if src == IP_ADDR_ANY, the IP address of the netif used to send is used as source address)
<i>dest</i>	the destination IP address to send the packet to
<i>ttl</i>	the TTL value to be set in the IP header
<i>tos</i>	the TOS value to be set in the IP header
<i>proto</i>	the PROTOCOL to be set in the IP header
<i>netif</i>	the netif on which to send this packet

Returns

ERR_OK if the packet was sent OK ERR_BUF if p doesn't have enough space for IP/LINK headers returns errors returned by netif->output

Note

ip_id: RFC791 "some host may be able to simply use unique identifiers independent of destination"

Definition at line 641 of file ip.c.

5.18.3.4 `struct netif* ip_route (ip_addr_t * dest)`

Finds the appropriate network interface for a given IP address. It searches the list of network interfaces linearly. A match is found if the masked IP address of the network interface equals the masked IP address given to the function.

Parameters

<i>dest</i>	the destination IP address for which to find the route
-------------	--

Returns

the netif on which to send to reach dest

Definition at line 124 of file ip.c.

5.18.4 Variable Documentation

5.18.4.1 `const struct ip_hdr* current_header`

Header of the input packet currently being processed.

Definition at line 105 of file ip.c.

5.18.4.2 `ip_addr_t current_iphdr_dest`

Destination IP address of current_header

Definition at line 109 of file ip.c.

5.18.4.3 `ip_addr_t current_iphdr_src`

Source IP address of current_header

Definition at line 107 of file ip.c.

5.18.4.4 `struct netif* current_netif`

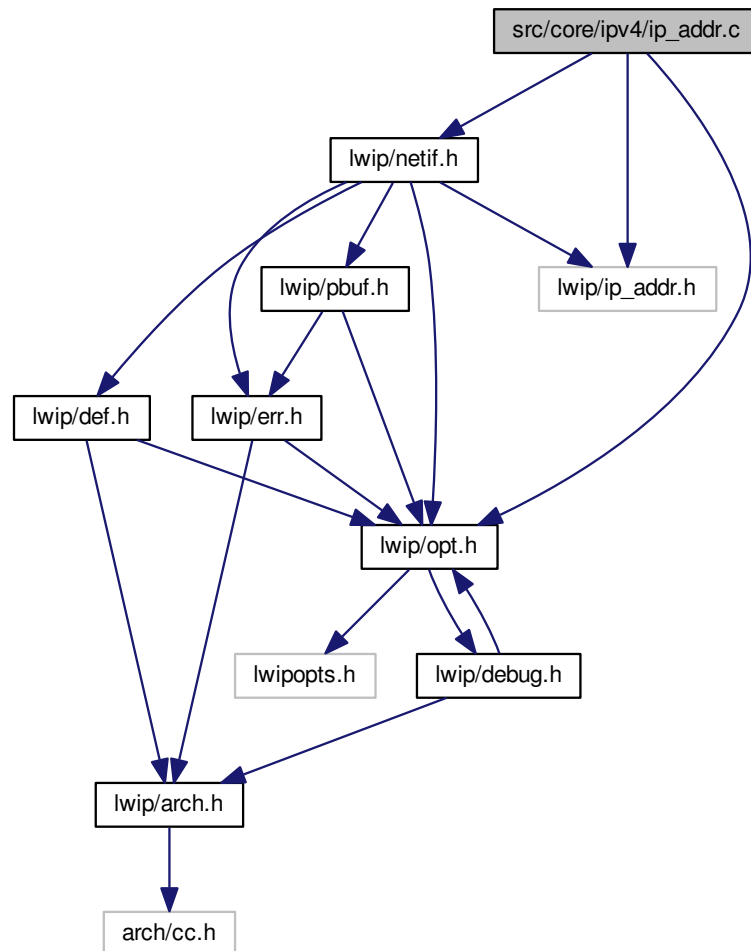
The interface that provided the packet for the current callback invocation.

Definition at line 100 of file ip.c.

5.19 src/core/ipv4/ip_addr.c File Reference

```
#include "lwip/opt.h"
#include "lwip/ip_addr.h"
#include "lwip/netif.h"
```

Include dependency graph for ip_addr.c:



Macros

- `#define in_range(c, lo, up) ((u8_t)c >= lo && (u8_t)c <= up)`
- `#define isprint(c) in_range(c, 0x20, 0x7f)`
- `#define isdigit(c) in_range(c, '0', '9')`
- `#define isxdigit(c) (isdigit(c) || in_range(c, 'a', 'f') || in_range(c, 'A', 'F'))`
- `#define islower(c) in_range(c, 'a', 'z')`
- `#define isspace(c) (c == ' ' || c == '\f' || c == '\n' || c == '\r' || c == '\t' || c == '\v')`

Functions

- `u8_t ip4_addr_isbroadcast(u32_t addr, const struct netif *netif)`
- `u8_t ip4_addr_netmask_valid(u32_t netmask)`
- `u32_t ipaddr_addr(const char *cp)`
- `int ipaddr_aton(const char *cp, ip_addr_t *addr)`
- `char * ipaddr_ntoa(const ip_addr_t *addr)`
- `char * ipaddr_ntoa_r(const ip_addr_t *addr, char *buf, int buflen)`

Variables

- const `ip_addr_t ip_addr_any` = { `IPADDR_ANY` }
- const `ip_addr_t ip_addr_broadcast` = { `IPADDR_BROADCAST` }

5.19.1 Detailed Description

This is the IPv4 address tools implementation.

5.19.2 Macro Definition Documentation

5.19.2.1 `#define in_range(c, lo, up) ((u8_t)c >= lo && (u8_t)c <= up)`

Definition at line 114 of file `ip_addr.c`.

5.19.2.2 `#define isdigit(c) in_range(c, '0', '9')`

Definition at line 116 of file `ip_addr.c`.

5.19.2.3 `#define islower(c) in_range(c, 'a', 'z')`

Definition at line 118 of file `ip_addr.c`.

5.19.2.4 `#define isprint(c) in_range(c, 0x20, 0x7f)`

Definition at line 115 of file `ip_addr.c`.

5.19.2.5 `#define isspace(c) (c == ' ' || c == '\f' || c == '\n' || c == '\r' || c == '\t' || c == '\v')`

Definition at line 119 of file `ip_addr.c`.

5.19.2.6 `#define isxdigit(c) (isdigit(c) || in_range(c, 'a', 'f') || in_range(c, 'A', 'F'))`

Definition at line 117 of file `ip_addr.c`.

5.19.3 Function Documentation

5.19.3.1 `u8_t ip4_addr_isbroadcast (u32_t addr, const struct netif * netif)`

Determine if an address is a broadcast address on a network interface

Parameters

<i>addr</i>	address to be checked
<i>netif</i>	the network interface against which the address is checked

Returns

returns non-zero if the address is a broadcast address

Definition at line 55 of file `ip_addr.c`.

5.19.3.2 `u8_t ip4_addr_netmask_valid (u32_t netmask)`

Checks if a netmask is valid (starting with ones, then only zeros)

Parameters

<i>netmask</i>	the IPv4 netmask to check (in network byte order!)
----------------	--

Returns

1 if the netmask is valid, 0 if it is not

Definition at line 90 of file ip_addr.c.

5.19.3.3 u32_t ipaddr_addr (const char * cp)

Ascii internet address interpretation routine. The value returned is in network order.

Parameters

<i>cp</i>	IP address in ascii representation (e.g. "127.0.0.1")
-----------	---

Returns

ip address in network order

Definition at line 130 of file ip_addr.c.

5.19.3.4 int ipaddr_aton (const char * cp, ip_addr_t * addr)

Check whether "cp" is a valid ascii representation of an Internet address and convert to a binary address. Returns 1 if the address is valid, 0 if not. This replaces inet_addr, the return value from which cannot distinguish between failure and a local broadcast address.

Parameters

<i>cp</i>	IP address in ascii representation (e.g. "127.0.0.1")
<i>addr</i>	pointer to which to save the ip address in network order

Returns

1 if cp could be converted to addr, 0 on failure

Definition at line 152 of file ip_addr.c.

5.19.3.5 char* ipaddr_ntoa (const ip_addr_t * addr)

Convert numeric IP address into decimal dotted ASCII representation. returns ptr to static buffer; not reentrant!

Parameters

<i>addr</i>	ip address in network order to convert
-------------	--

Returns

pointer to a global static (!) buffer that holds the ASCII representation of addr

Definition at line 261 of file ip_addr.c.

5.19.3.6 char* ipaddr_ntoa_r (const ip_addr_t * addr, char * buf, int buflen)

Same as ipaddr_ntoa, but reentrant since a user-supplied buffer is used.

Parameters

<i>addr</i>	ip address in network order to convert
<i>buf</i>	target buffer where the string is stored
<i>buflen</i>	length of buf

Returns

either pointer to buf which now holds the ASCII representation of addr or NULL if buf was too small

Definition at line 276 of file ip_addr.c.

5.19.4 Variable Documentation

5.19.4.1 `const ip_addr_t ip_addr_any = { IPADDR_ANY }`

Definition at line 44 of file ip_addr.c.

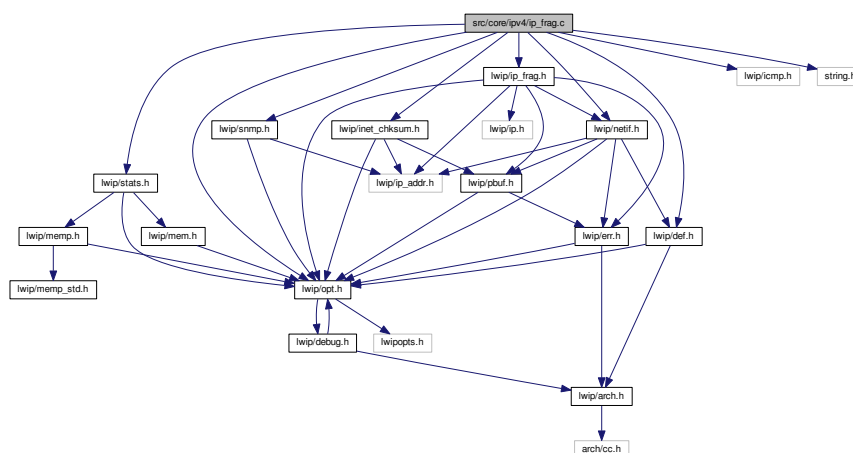
5.19.4.2 `const ip_addr_t ip_addr_broadcast = { IPADDR_BROADCAST }`

Definition at line 45 of file ip_addr.c.

5.20 `src/core/ipv4/ip_frag.c` File Reference

```
#include "lwip/opt.h"
#include "lwip/ip_frag.h"
#include "lwip/def.h"
#include "lwip/inet_chksum.h"
#include "lwip/netif.h"
#include "lwip/snmp.h"
#include "lwip/stats.h"
#include "lwip/icmp.h"
#include <string.h>
```

Include dependency graph for ip_frag.c:



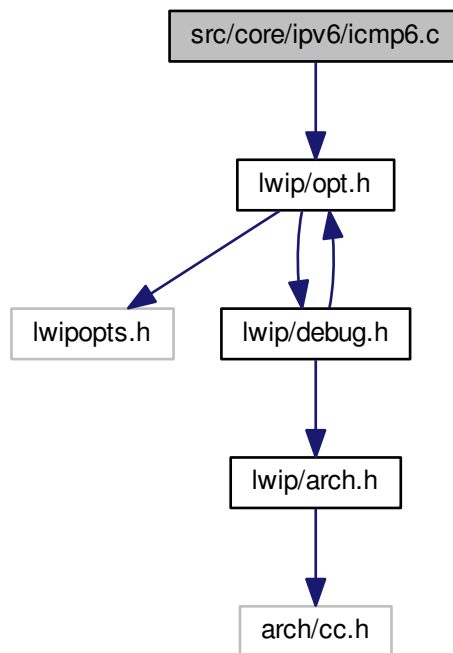
5.20.1 Detailed Description

This is the IPv4 packet segmentation and reassembly implementation.

5.21 src/core/ipv6/icmp6.c File Reference

```
#include "lwip/opt.h"
```

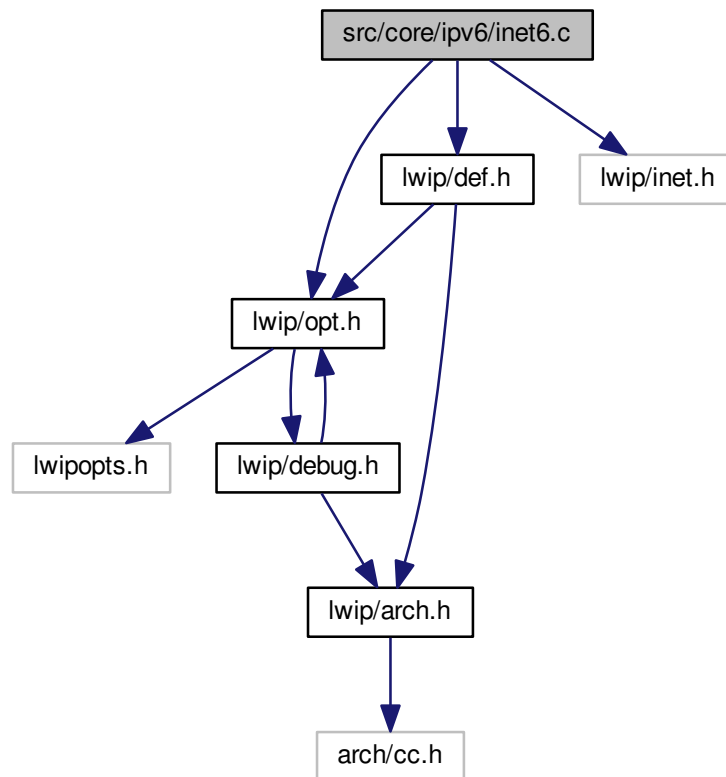
Include dependency graph for icmp6.c:



5.22 src/core/ipv6/inet6.c File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/inet.h"
```

Include dependency graph for inet6.c:



Functions

- `u16_t inet_chksum_pseudo` (struct `pbuf` *p, struct `ip_addr` *src, struct `ip_addr` *dest, u8_t proto, u32_t proto_len)
- `u16_t inet_chksum` (void *dataptr, u16_t len)
- `u16_t inet_chksum_pbuf` (struct `pbuf` *p)

5.22.1 Detailed Description

Functions common to all TCP/IPv6 modules, such as the Internet checksum and the byte order functions.

5.22.2 Function Documentation

5.22.2.1 `u16_t inet_chksum (void * dataptr, u16_t len)`

Definition at line 129 of file `inet6.c`.

5.22.2.2 `u16_t inet_chksum_pbuf (struct pbuf * p)`

Calculate a checksum over a chain of pbufs (without pseudo-header, much like `inet_chksum` only pbufs are used).

Parameters

<i>p</i>	pbuff chain over that the checksum should be calculated
----------	---

Returns

checksum (as u16_t) to be saved directly in the protocol header

Definition at line 140 of file inet6.c.

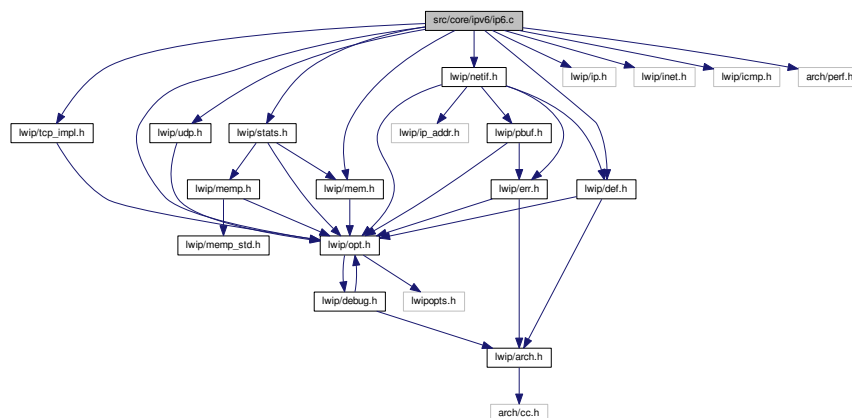
5.22.2.3 u16_t inet_chksum_pseudo (struct pbuf * *p*, struct ip_addr * *src*, struct ip_addr * *dest*, u8_t *proto*, u32_t *proto_len*)

Definition at line 80 of file inet6.c.

5.23 src/core/ipv6/ip6.c File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/mem.h"
#include "lwip/ip.h"
#include "lwip/inet.h"
#include "lwip/netif.h"
#include "lwip/icmp.h"
#include "lwip/udp.h"
#include "lwip/tcp_impl.h"
#include "lwip/stats.h"
#include "arch/perf.h"
```

Include dependency graph for ip6.c:



Functions

- void `ip_init` (void)
- struct `netif` * `ip_route` (struct `ip_addr` **dest*)
- void `ip_input` (struct `pbuf` **p*, struct `netif` **inp*)
- `err_t` `ip_output_if` (struct `pbuf` **p*, struct `ip_addr` **src*, struct `ip_addr` **dest*, u8_t *tll*, u8_t *proto*, struct `netif` **netif*)
- `err_t` `ip_output` (struct `pbuf` **p*, struct `ip_addr` **src*, struct `ip_addr` **dest*, u8_t *tll*, u8_t *proto*)

5.23.1 Function Documentation

5.23.1.1 void ip_init (void)

Definition at line 63 of file ip6.c.

5.23.1.2 void ip_input (struct pbuf * *p*, struct netif * *inp*)

This function is called by the network interface device driver when an IP packet is received. The function does the basic checks of the IP header such as packet size being at least larger than the header size etc. If the packet was not destined for us, the packet is forwarded (using ip_forward). The IP checksum is always checked.

Finally, the packet is sent to the upper layer protocol input function.

Parameters

<i>p</i>	the received IP packet (p->payload points to IP header)
<i>inp</i>	the netif on which this packet was received

Returns

ERR_OK if the packet was processed (could return ERR_* if it wasn't processed, but currently always returns ERR_OK)

Definition at line 157 of file ip6.c.

5.23.1.3 err_t ip_output (struct pbuf * *p*, struct ip_addr * *src*, struct ip_addr * *dest*, u8_t *tll*, u8_t *proto*)

Definition at line 317 of file ip6.c.

5.23.1.4 err_t ip_output_if (struct pbuf * *p*, struct ip_addr * *src*, struct ip_addr * *dest*, u8_t *tll*, u8_t *proto*, struct netif * *netif*)

Definition at line 260 of file ip6.c.

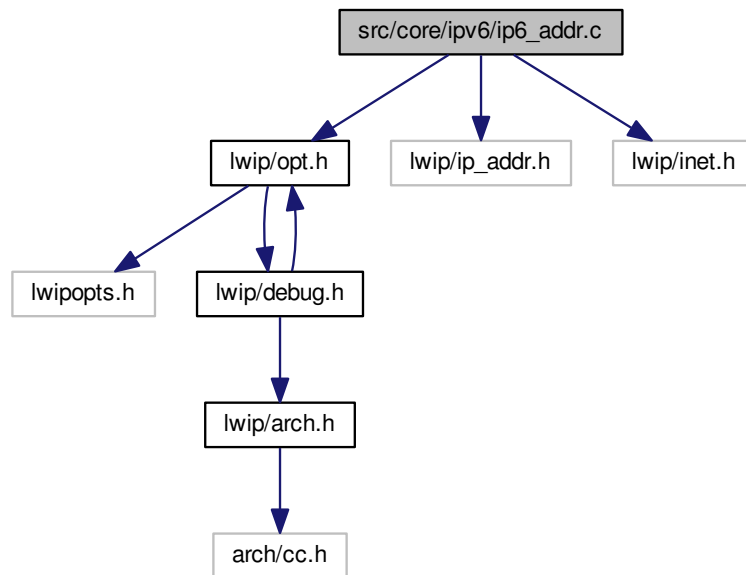
5.23.1.5 struct netif* ip_route (struct ip_addr * *dest*)

Definition at line 75 of file ip6.c.

5.24 src/core/ipv6/ip6_addr.c File Reference

```
#include "lwip/opt.h"
#include "lwip/ip_addr.h"
#include "lwip/inet.h"
```


Include dependency graph for ip6_addr.c:



Functions

- `u8_t ip_addr_netcmp` (struct `ip_addr` *`addr1`, struct `ip_addr` *`addr2`, struct `ip_addr` *`mask`)
- `u8_t ip_addr_cmp` (struct `ip_addr` *`addr1`, struct `ip_addr` *`addr2`)
- `void ip_addr_set` (struct `ip_addr` *`dest`, struct `ip_addr` *`src`)
- `u8_t ip_addr_isany` (struct `ip_addr` *`addr`)

5.24.1 Function Documentation

5.24.1.1 `u8_t ip_addr_cmp (struct ip_addr * addr1, struct ip_addr * addr2)`

Definition at line 49 of file `ip6_addr.c`.

5.24.1.2 `u8_t ip_addr_isany (struct ip_addr * addr)`

Definition at line 68 of file `ip6_addr.c`.

5.24.1.3 `u8_t ip_addr_netcmp (struct ip_addr * addr1, struct ip_addr * addr2, struct ip_addr * mask)`

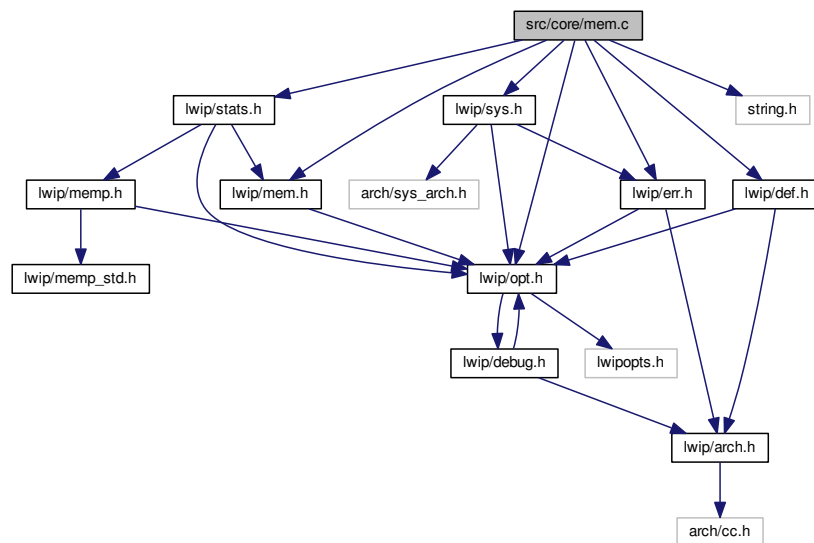
Definition at line 38 of file `ip6_addr.c`.

5.24.1.4 `void ip_addr_set (struct ip_addr * dest, struct ip_addr * src)`

Definition at line 58 of file `ip6_addr.c`.

5.25 src/core/mem.c File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/mem.h"
#include "lwip/sys.h"
#include "lwip/stats.h"
#include "lwip/err.h"
#include <string.h>
Include dependency graph for mem.c:
```



Data Structures

- struct [mem](#)

Macros

- `#define MIN_SIZE 12`
- `#define MIN_SIZE_ALIGNED LWIP_MEM_ALIGN_SIZE(MIN_SIZE)`
- `#define SIZEOF_STRUCT_MEM LWIP_MEM_ALIGN_SIZE(sizeof(struct mem))`
- `#define MEM_SIZE_ALIGNED LWIP_MEM_ALIGN_SIZE(MEM_SIZE)`
- `#define LWIP_RAM_HEAP_POINTER ram_heap`
- `#define LWIP_MEM_FREE_DECL_PROTECT()`
- `#define LWIP_MEM_FREE_PROTECT() sys_mutex_lock(&mem_mutex)`
- `#define LWIP_MEM_FREE_UNPROTECT() sys_mutex_unlock(&mem_mutex)`
- `#define LWIP_MEM_ALLOC_DECL_PROTECT()`
- `#define LWIP_MEM_ALLOC_PROTECT()`
- `#define LWIP_MEM_ALLOC_UNPROTECT()`

Functions

- void [mem_init](#) (void)

- void [mem_free](#) (void *rmem)
- void * [mem_trim](#) (void *rmem, [mem_size_t](#) newsize)
- void * [mem_malloc](#) ([mem_size_t](#) size)
- void * [mem_calloc](#) ([mem_size_t](#) count, [mem_size_t](#) size)

Variables

- [u8_t ram_heap](#) [[MEM_SIZE_ALIGNED](#)+(2 *[SIZEOF_STRUCT_MEM](#))+[MEM_ALIGNMENT](#)]

5.25.1 Detailed Description

Dynamic memory manager

This is a lightweight replacement for the standard C library malloc().

If you want to use the standard C library malloc() instead, define MEM_LIBC_MALLOC to 1 in your lwipopts.h

To let [mem_malloc\(\)](#) use pools (prevents fragmentation and is much faster than a heap but might waste some memory), define MEM_USE_POOLS to 1, define MEM_USE_CUSTOM_POOLS to 1 and create a file "lwippools.h" that includes a list of pools like this (more pools can be added between _START and _END):

Define three pools with sizes 256, 512, and 1512 bytes LWIP_MALLOC_MEMPOOL_START [LWIP_MALLOC_MEMPOOL\(20, 256\)](#) [LWIP_MALLOC_MEMPOOL\(10, 512\)](#) [LWIP_MALLOC_MEMPOOL\(5, 1512\)](#) [LWIP_MALLOC_MEMPOOL_END](#)

5.25.2 Macro Definition Documentation

5.25.2.1 #define LWIP_MEM_ALLOC_DECL_PROTECT()

Definition at line 217 of file mem.c.

5.25.2.2 #define LWIP_MEM_ALLOC_PROTECT()

Definition at line 218 of file mem.c.

5.25.2.3 #define LWIP_MEM_ALLOC_UNPROTECT()

Definition at line 219 of file mem.c.

5.25.2.4 #define LWIP_MEM_FREE_DECL_PROTECT()

Definition at line 213 of file mem.c.

5.25.2.5 #define LWIP_MEM_FREE_PROTECT() sys_mutex_lock(&mem_mutex)

Definition at line 214 of file mem.c.

5.25.2.6 #define LWIP_MEM_FREE_UNPROTECT() sys_mutex_unlock(&mem_mutex)

Definition at line 215 of file mem.c.

5.25.2.7 `#define LWIP_RAM_HEAP_POINTER ram_heap`

Definition at line 183 of file mem.c.

5.25.2.8 `#define MEM_SIZE_ALIGNED LWIP_MEM_ALIGN_SIZE(MEM_SIZE)`

Definition at line 174 of file mem.c.

5.25.2.9 `#define MIN_SIZE 12`

All allocated blocks will be MIN_SIZE bytes big, at least! MIN_SIZE can be overridden to suit your needs. Smaller values save space, larger values could prevent too small blocks to fragment the RAM too much.

Definition at line 169 of file mem.c.

5.25.2.10 `#define MIN_SIZE_ALIGNED LWIP_MEM_ALIGN_SIZE(MIN_SIZE)`

Definition at line 172 of file mem.c.

5.25.2.11 `#define SIZEOF_STRUCT_MEM LWIP_MEM_ALIGN_SIZE(sizeof(struct mem))`

Definition at line 173 of file mem.c.

5.25.3 Function Documentation

5.25.3.1 `void* mem_malloc (mem_size_t count, mem_size_t size)`

Contiguously allocates enough space for count objects that are size bytes of memory each and returns a pointer to the allocated memory.

The allocated memory is filled with bytes of value zero.

Parameters

<i>count</i>	number of objects to allocate
<i>size</i>	size of the objects to allocate

Returns

pointer to allocated memory / NULL pointer if there is an error

Definition at line 646 of file mem.c.

5.25.3.2 `void mem_free (void * rmem)`

Put a struct mem back on the heap

Parameters

<i>rmem</i>	is the data portion of a struct mem as returned by a previous call to mem_malloc()
-------------	--

Definition at line 311 of file mem.c.

5.25.3.3 void mem_init (void)

Zero the heap and initialize start, end and lowest-free

Definition at line 274 of file mem.c.

5.25.3.4 void* mem_malloc (mem_size_t size)

Adam's [mem_malloc\(\)](#) plus solution for bug #17922 Allocate a block of memory with a minimum of 'size' bytes.

Parameters

<i>size</i>	is the minimum size of the requested block in bytes.
-------------	--

Returns

pointer to allocated memory or NULL if no free memory was found.

Note that the returned value will always be aligned (as defined by MEM_ALIGNMENT).

Definition at line 494 of file mem.c.

5.25.3.5 void* mem_trim (void * rmem, mem_size_t newsize)

Shrink memory returned by [mem_malloc\(\)](#).

Parameters

<i>rmem</i>	pointer to memory allocated by mem_malloc the is to be shrinked
<i>newsize</i>	required size after shrinking (needs to be smaller than or equal to the previous size)

Returns

for compatibility reasons: is always == rmem, at the moment or NULL if newsize is > old size, in which case rmem is NOT touched or freed!

Definition at line 369 of file mem.c.

5.25.4 Variable Documentation

5.25.4.1 u8_t ram_heap[MEM_SIZE_ALIGNED+(2*SIZEOF_STRUCT_MEM)+MEM_ALIGNMENT]

If you want to relocate the heap to external memory, simply define LWIP_RAM_HEAP_POINTER as a void-pointer to that location. If so, make sure the memory at that location is big enough (see below on how that space is calculated). the heap. we need one struct mem at the end and some room for alignment

Definition at line 182 of file mem.c.

5.26 src/core/memp.c File Reference

```
#include "lwip/opt.h"
```


lwIP has dedicated pools for many structures (netconn, protocol control blocks, packet buffers, ...). All these pools are managed here.

5.26.2 Macro Definition Documentation

5.26.2.1 `#define LWIP_MEMPOOL(name, num, size, desc) LWIP_MEM_ALIGN_SIZE(size),`

5.26.2.2 `#define LWIP_MEMPOOL(name, num, size, desc) (num),`

5.26.2.3 `#define LWIP_MEMPOOL(name, num, size, desc) + ((num) * (MEMP_SIZE + MEMP_ALIGN_SIZE(size)))`

5.26.2.4 `#define MEMP_ALIGN_SIZE(x) (LWIP_MEM_ALIGN_SIZE(x))`

Definition at line 111 of file memp.c.

5.26.2.5 `#define MEMP_SIZE 0`

Definition at line 110 of file memp.c.

5.26.3 Function Documentation

5.26.3.1 `void memp_free (memp_t type, void * mem)`

Put an element back into its pool.

Parameters

<i>type</i>	the pool where to put mem
<i>mem</i>	the memp element to free

Definition at line 435 of file memp.c.

5.26.3.2 `void memp_init (void)`

Initialize this module.

Carves out memp_memory into linked lists for each pool-type.

Definition at line 338 of file memp.c.

5.26.3.3 `void* memp_malloc (memp_t type)`

Get an element from a specific pool.

Parameters

<i>type</i>	the pool to get an element from
-------------	---------------------------------

the debug version has two more parameters:

Parameters

<i>file</i>	file name calling this function
<i>line</i>	number of line where this function is called

Returns

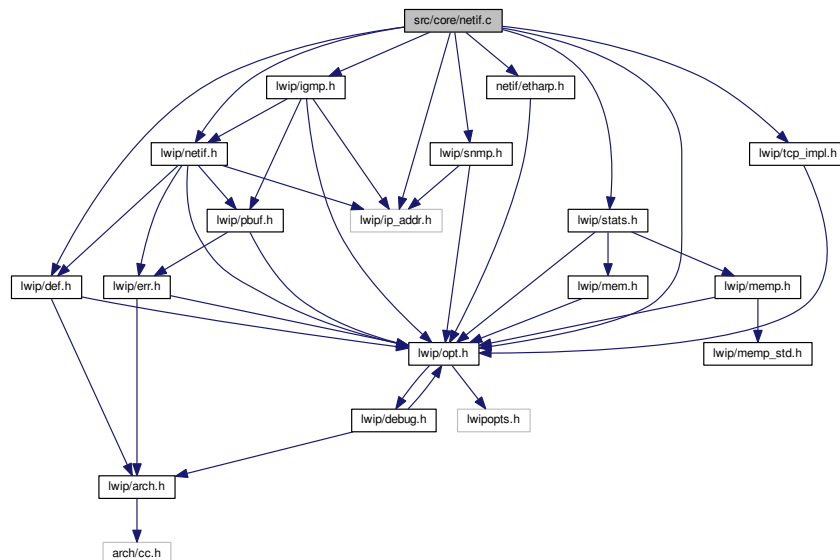
a pointer to the allocated memory or a NULL pointer on error

Definition at line 390 of file memp.c.

5.27 src/core/netif.c File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/ip_addr.h"
#include "lwip/netif.h"
#include "lwip/tcp_impl.h"
#include "lwip/snmp.h"
#include "lwip/igmp.h"
#include "netif/etharp.h"
#include "lwip/stats.h"
```

Include dependency graph for netif.c:

**Macros**

- `#define NETIF_STATUS_CALLBACK(n)`
- `#define NETIF_LINK_CALLBACK(n)`

Functions

- void `netif_init` (void)
- struct `netif` * `netif_add` (struct `netif` *`netif`, `ip_addr_t` *`ipaddr`, `ip_addr_t` *`netmask`, `ip_addr_t` *`gw`, void *`state`, `netif_init_fn` `init`, `netif_input_fn` `input`)
- void `netif_set_addr` (struct `netif` *`netif`, `ip_addr_t` *`ipaddr`, `ip_addr_t` *`netmask`, `ip_addr_t` *`gw`)
- void `netif_remove` (struct `netif` *`netif`)
- struct `netif` * `netif_find` (char *`name`)
- void `netif_set_ipaddr` (struct `netif` *`netif`, `ip_addr_t` *`ipaddr`)

- void `netif_set_gw` (struct `netif` *`netif`, `ip_addr_t` *`gw`)
- void `netif_set_netmask` (struct `netif` *`netif`, `ip_addr_t` *`netmask`)
- void `netif_set_default` (struct `netif` *`netif`)
- void `netif_set_up` (struct `netif` *`netif`)
- void `netif_set_down` (struct `netif` *`netif`)
- void `netif_set_link_up` (struct `netif` *`netif`)
- void `netif_set_link_down` (struct `netif` *`netif`)

Variables

- struct `netif` * `netif_list`
- struct `netif` * `netif_default`

5.27.1 Detailed Description

LwIP network interface abstraction

5.27.2 Macro Definition Documentation

5.27.2.1 #define NETIF_LINK_CALLBACK(n)

Definition at line 72 of file `netif.c`.

5.27.2.2 #define NETIF_STATUS_CALLBACK(n)

Definition at line 66 of file `netif.c`.

5.27.3 Function Documentation

5.27.3.1 struct netif* netif_add (struct netif * *netif*, ip_addr_t * *ipaddr*, ip_addr_t * *netmask*, ip_addr_t * *gw*, void * *state*, netif_init_fn *init*, netif_input_fn *input*)

Add a network interface to the list of LwIP netifs.

Parameters

<i>netif</i>	a pre-allocated netif structure
<i>ipaddr</i>	IP address for the new netif
<i>netmask</i>	network mask for the new netif
<i>gw</i>	default gateway IP address for the new netif
<i>state</i>	opaque data passed to the new netif
<i>init</i>	callback function that initializes the interface
<i>input</i>	callback function that is called to pass ingress packets up in the protocol layer stack.

Returns

`netif`, or NULL if failed.

Definition at line 139 of file `netif.c`.

5.27.3.2 struct netif* netif_find (char * *name*)

Find a network interface by searching for its name

Parameters

<i>name</i>	the name of the netif (like netif->name) plus concatenated number in ascii representation (e.g. 'en0')
-------------	--

Definition at line 290 of file netif.c.

5.27.3.3 void netif_init (void)

Definition at line 106 of file netif.c.

5.27.3.4 void netif_remove (struct netif * netif)

Remove a network interface from the list of lwIP netifs.

Parameters

<i>netif</i>	the network interface to remove
--------------	---------------------------------

Definition at line 235 of file netif.c.

5.27.3.5 void netif_set_addr (struct netif * netif, ip_addr_t * ipaddr, ip_addr_t * netmask, ip_addr_t * gw)

Change IP address configuration for a network interface (including netmask and default gateway).

Parameters

<i>netif</i>	the network interface to change
<i>ipaddr</i>	the new IP address
<i>netmask</i>	the new netmask
<i>gw</i>	the new default gateway

Definition at line 221 of file netif.c.

5.27.3.6 void netif_set_default (struct netif * netif)

Set a network interface as the default network interface (used to output all packets for which no specific route is found)

Parameters

<i>netif</i>	the default network interface
--------------	-------------------------------

Definition at line 430 of file netif.c.

5.27.3.7 void netif_set_down (struct netif * netif)

Bring an interface down, disabling any traffic processing.

Note

: Enabling DHCP on a down interface will make it come up once configured.

See also

dhcp_start()

Definition at line 490 of file netif.c.

5.27.3.8 void netif_set_gw (struct netif * *netif*, ip_addr_t * *gw*)

Change the default gateway for a network interface

Parameters

<i>netif</i>	the network interface to change
<i>gw</i>	the new default gateway

Note

call [netif_set_addr\(\)](#) if you also want to change ip address and netmask

Definition at line 388 of file netif.c.

5.27.3.9 void [netif_set_ipaddr](#) (struct netif * *netif*, ip_addr_t * *ipaddr*)

Change the IP address of a network interface

Parameters

<i>netif</i>	the network interface to change
<i>ipaddr</i>	the new IP address

Note

call [netif_set_addr\(\)](#) if you also want to change netmask and default gateway

Definition at line 323 of file netif.c.

5.27.3.10 void [netif_set_link_down](#) (struct netif * *netif*)

Called by a driver when its link goes down

Definition at line 574 of file netif.c.

5.27.3.11 void [netif_set_link_up](#) (struct netif * *netif*)

Called by a driver when its link goes up

Definition at line 535 of file netif.c.

5.27.3.12 void [netif_set_netmask](#) (struct netif * *netif*, ip_addr_t * *netmask*)

Change the netmask of a network interface

Parameters

<i>netif</i>	the network interface to change
<i>netmask</i>	the new netmask

Note

call [netif_set_addr\(\)](#) if you also want to change ip address and default gateway

Definition at line 409 of file netif.c.

5.27.3.13 void [netif_set_up](#) (struct netif * *netif*)

Bring an interface up, available for processing traffic.

Note

: Enabling DHCP on a down interface will make it come up once configured.

See also

dhcp_start()

Definition at line 453 of file netif.c.

5.27.4 Variable Documentation**5.27.4.1 struct netif* netif_default**

The default network interface.

Definition at line 76 of file netif.c.

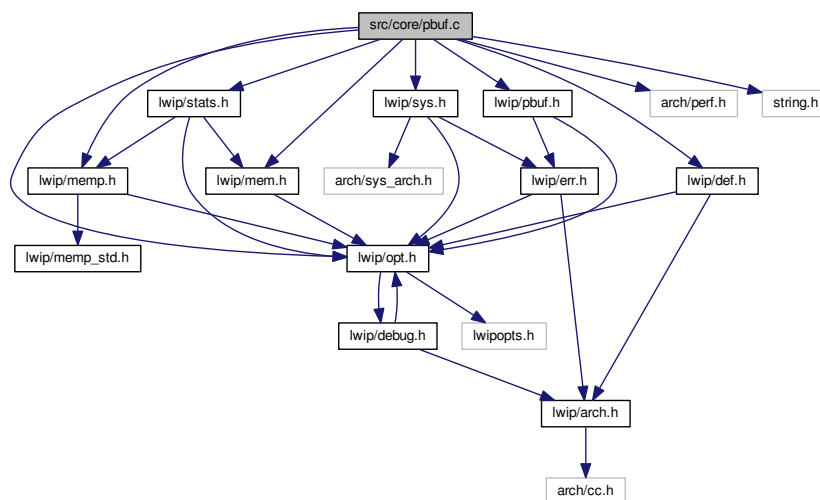
5.27.4.2 struct netif* netif_list

The list of network interfaces.

Definition at line 75 of file netif.c.

5.28 src/core/pbuf.c File Reference

```
#include "lwip/opt.h"
#include "lwip/stats.h"
#include "lwip/def.h"
#include "lwip/mem.h"
#include "lwip/memp.h"
#include "lwip/pbuf.h"
#include "lwip/sys.h"
#include "arch/perf.h"
#include <string.h>
Include dependency graph for pbuf.c:
```



Macros

- `#define SIZEOF_STRUCT_PBUF LWIP_MEM_ALIGN_SIZE(sizeof(struct pbuf))`
- `#define PBUF_POOL_BUFSIZE_ALIGNED LWIP_MEM_ALIGN_SIZE(PBUF_POOL_BUFSIZE)`
- `#define PBUF_POOL_IS_EMPTY()`

Functions

- `struct pbuf * pbuf_alloc (pbuf_layer layer, u16_t length, pbuf_type type)`
- `void pbuf_realloc (struct pbuf *p, u16_t new_len)`
- `u8_t pbuf_header (struct pbuf *p, s16_t header_size_increment)`
- `u8_t pbuf_free (struct pbuf *p)`
- `u8_t pbuf_clen (struct pbuf *p)`
- `void pbuf_ref (struct pbuf *p)`
- `void pbuf_cat (struct pbuf *h, struct pbuf *t)`
- `void pbuf_chain (struct pbuf *h, struct pbuf *t)`
- `struct pbuf * pbuf_dechain (struct pbuf *p)`
- `err_t pbuf_copy (struct pbuf *p_to, struct pbuf *p_from)`
- `u16_t pbuf_copy_partial (struct pbuf *buf, void *dataptr, u16_t len, u16_t offset)`
- `err_t pbuf_take (struct pbuf *buf, const void *dataptr, u16_t len)`
- `struct pbuf * pbuf_coalesce (struct pbuf *p, pbuf_layer layer)`
- `u8_t pbuf_get_at (struct pbuf *p, u16_t offset)`
- `u16_t pbuf_memcmp (struct pbuf *p, u16_t offset, const void *s2, u16_t n)`
- `u16_t pbuf_memfind (struct pbuf *p, const void *mem, u16_t mem_len, u16_t start_offset)`
- `u16_t pbuf_strstr (struct pbuf *p, const char *substr)`

5.28.1 Detailed Description

Packet buffer management

Packets are built from the pbuf data structure. It supports dynamic memory allocation for packet contents or can reference externally managed packet contents both in RAM and ROM. Quick allocation for incoming packets is provided through pools with fixed sized pbufs.

A packet may span over multiple pbufs, chained as a singly linked list. This is called a "pbuf chain".

Multiple packets may be queued, also using this singly linked list. This is called a "packet queue".

So, a packet queue consists of one or more pbuf chains, each of which consist of one or more pbufs. CURRENTLY, PACKET QUEUES ARE NOT SUPPORTED!!! Use helper structs to queue multiple packets.

The differences between a pbuf chain and a packet queue are very precise but subtle.

The last pbuf of a packet has a `->tot_len` field that equals the `->len` field. It can be found by traversing the list. If the last pbuf of a packet has a `->next` field other than NULL, more packets are on the queue.

Therefore, looping through a pbuf of a single packet, has an loop end condition (`tot_len == p->len`), NOT (`next == NULL`).

5.28.2 Macro Definition Documentation

5.28.2.1 `#define PBUF_POOL_BUFSIZE_ALIGNED LWIP_MEM_ALIGN_SIZE(PBUF_POOL_BUFSIZE)`

Definition at line 85 of file pbuf.c.

5.28.2.2 `#define PBUF_POOL_IS_EMPTY()`

Definition at line 88 of file pbuf.c.

5.28.2.3 `#define SIZEOF_STRUCT_PBUF LWIP_MEM_ALIGN_SIZE(sizeof(struct pbuf))`

Definition at line 82 of file pbuf.c.

5.28.3 Function Documentation

5.28.3.1 `struct pbuf* pbuf_alloc (pbuf_layer layer, u16_t length, pbuf_type type)`

Allocates a pbuf of the given type (possibly a chain for PBUF_POOL type).

The actual memory allocated for the pbuf is determined by the layer at which the pbuf is allocated and the requested size (from the size parameter).

Parameters

<i>layer</i>	flag to define header size
<i>length</i>	size of the pbuf's payload
<i>type</i>	this parameter decides how and where the pbuf should be allocated as follows:

- PBUF_RAM: buffer memory for pbuf is allocated as one large chunk. This includes protocol headers as well.
- PBUF_ROM: no buffer memory is allocated for the pbuf, even for protocol headers. Additional headers must be prepended by allocating another pbuf and chain in to the front of the ROM pbuf. It is assumed that the memory used is really similar to ROM in that it is immutable and will not be changed. Memory which is dynamic should generally not be attached to PBUF_ROM pbufs. Use PBUF_REF instead.
- PBUF_REF: no buffer memory is allocated for the pbuf, even for protocol headers. It is assumed that the pbuf is only being used in a single thread. If the pbuf gets queued, then pbuf_take should be called to copy the buffer.
- PBUF_POOL: the pbuf is allocated as a pbuf chain, with pbufs from the pbuf pool that is allocated during [pbuf_init\(\)](#).

Returns

the allocated pbuf. If multiple pbufs where allocated, this is the first pbuf of a pbuf chain.

Definition at line 207 of file pbuf.c.

5.28.3.2 `void pbuf_cat (struct pbuf * h, struct pbuf * t)`

Concatenate two pbufs (each may be a pbuf chain) and take over the caller's reference of the tail pbuf.

Note

The caller MAY NOT reference the tail pbuf afterwards. Use [pbuf_chain\(\)](#) for that purpose.

See also

[pbuf_chain\(\)](#)

Definition at line 745 of file pbuf.c.

5.28.3.3 `void pbuf_chain (struct pbuf * h, struct pbuf * t)`

Chain two pbufs (or pbuf chains) together.

The caller MUST call pbuf_free(t) once it has stopped using it. Use [pbuf_cat\(\)](#) instead if you no longer use t.

Parameters

<i>h</i>	head pbuf (chain)
<i>t</i>	tail pbuf (chain)

Note

The pbufs MUST belong to the same packet.
MAY NOT be called on a packet queue.

The ->tot_len fields of all pbufs of the head chain are adjusted. The ->next field of the last pbuf of the head chain is adjusted. The ->ref field of the first pbuf of the tail chain is adjusted.

Definition at line 786 of file pbuf.c.

5.28.3.4 `u8_t pbuf_clen (struct pbuf * p)`

Count number of pbufs in a chain

Parameters

<i>p</i>	first pbuf of chain
----------	---------------------

Returns

the number of pbufs in a chain

Definition at line 704 of file pbuf.c.

5.28.3.5 `struct pbuf* pbuf_coalesce (struct pbuf * p, pbuf_layer layer)`

Creates a single pbuf out of a queue of pbufs.

Remarks

: Either the source pbuf 'p' is freed by this function or the original pbuf 'p' is returned, therefore the caller has to check the result!

Parameters

<i>p</i>	the source pbuf
<i>layer</i>	pbuf_layer of the new pbuf

Returns

a new, single pbuf (p->next is NULL) or the old pbuf if allocation fails

Definition at line 1010 of file pbuf.c.

5.28.3.6 `err_t pbuf_copy (struct pbuf * p_to, struct pbuf * p_from)`

Create PBUF_RAM copies of pbufs.

Used to queue packets on behalf of the lwIP stack, such as ARP based queueing.

Note

You MUST explicitly use `p = pbuf_take(p);`
Only one packet is copied, no packet queue!

Parameters

<i>p_to</i>	pbuf destination of the copy
<i>p_from</i>	pbuf source of the copy

Returns

ERR_OK if pbuf was copied ERR_ARG if one of the pbufs is NULL or p_to is not big enough to hold p_from

Definition at line 852 of file pbuf.c.

5.28.3.7 u16_t pbuf_copy_partial (struct pbuf * buf, void * dataptr, u16_t len, u16_t offset)

Copy (part of) the contents of a packet buffer to an application supplied buffer.

Parameters

<i>buf</i>	the pbuf from which to copy data
<i>dataptr</i>	the application supplied buffer
<i>len</i>	length of data to copy (dataptr must be big enough). No more than buf->tot_len will be copied, irrespective of len
<i>offset</i>	offset into the packet buffer from where to begin copying len bytes

Returns

the number of bytes copied, or 0 on failure

Definition at line 918 of file pbuf.c.

5.28.3.8 struct pbuf* pbuf_dechain (struct pbuf * p)

Dechains the first pbuf from its succeeding pbufs in the chain.

Makes p->tot_len field equal to p->len.

Parameters

<i>p</i>	pbuf to dechain
----------	-----------------

Returns

remainder of the pbuf chain, or NULL if it was de-allocated.

Note

May not be called on a packet queue.

Definition at line 803 of file pbuf.c.

5.28.3.9 u8_t pbuf_free (struct pbuf * p)

Dereference a pbuf chain or queue and deallocate any no-longer-used pbufs at the head of this chain or queue.

Decrements the pbuf reference count. If it reaches zero, the pbuf is deallocated.

For a pbuf chain, this is repeated for each pbuf in the chain, up to the first pbuf which has a non-zero reference count after decrementing. So, when all reference counts are one, the whole chain is free'd.

Parameters

<i>p</i>	The pbuf (chain) to be dereferenced.
----------	--------------------------------------

Returns

the number of pbufs that were de-allocated from the head of the chain.

Note

MUST NOT be called on a packet queue (Not verified to work yet).
the reference counter of a pbuf equals the number of pointers that refer to the pbuf (or into the pbuf).

Definition at line 618 of file pbuf.c.

5.28.3.10 u8_t pbuf_get_at (struct pbuf * p, u16_t offset)

Get one byte from the specified position in a pbuf WARNING: returns zero for offset >= p->tot_len

Parameters

<i>p</i>	pbuf to parse
<i>offset</i>	offset into p of the byte to return

Returns

byte at an offset into p OR ZERO IF 'offset' >= p->tot_len

Definition at line 1077 of file pbuf.c.

5.28.3.11 u8_t pbuf_header (struct pbuf * p, s16_t header_size_increment)

Adjusts the payload pointer to hide or reveal headers in the payload.

Adjusts the ->payload pointer so that space for a header (dis)appears in the pbuf payload.

The ->payload, ->tot_len and ->len fields are adjusted.

Parameters

<i>p</i>	pbuf to change the header size.
<i>header_size_increment</i>	Number of bytes to increment header size which increases the size of the pbuf. New space is on the front. (Using a negative value decreases the header size.) If hdr_size_inc is 0, this function does nothing and returns succesful.

PBUF_ROM and PBUF_REF type buffers cannot have their sizes increased, so the call will fail. A check is made that the increase in header size does not move the payload pointer in front of the start of the buffer.

Returns

non-zero on failure, zero on success.

Definition at line 511 of file pbuf.c.

5.28.3.12 u16_t pbuf_memcmp (struct pbuf * p, u16_t offset, const void * s2, u16_t n)

Compare pbuf contents at specified offset with memory s2, both of length n

Parameters

<i>p</i>	pbuf to compare
<i>offset</i>	offset into p at which to start comparing
<i>s2</i>	buffer to compare
<i>n</i>	length of buffer to compare

Returns

zero if equal, nonzero otherwise (0xffff if p is too short, diff+offset+1 otherwise)

Definition at line 1104 of file pbuf.c.

5.28.3.13 u16_t pbuf_memfind (struct pbuf * p, const void * mem, u16_t mem_len, u16_t start_offset)

Find occurrence of mem (with length mem_len) in pbuf p, starting at offset start_offset.

Parameters

<i>p</i>	pbuf to search, maximum length is 0xFFFFE since 0xFFFF is used as return value 'not found'
<i>mem</i>	search for the contents of this buffer
<i>mem_len</i>	length of 'mem'
<i>start_offset</i>	offset into p at which to start searching

Returns

0xFFFF if substr was not found in p or the index where it was found

Definition at line 1140 of file pbuf.c.

5.28.3.14 void pbuf_realloc (struct pbuf * p, u16_t new_len)

Shrink a pbuf chain to a desired length.

Parameters

<i>p</i>	pbuf to shrink.
<i>new_len</i>	desired new length of pbuf chain

Depending on the desired length, the first few pbuffers in a chain might be skipped and left unchanged. The new last pbuf in the chain will be resized, and any remaining pbuffers will be freed.

Note

If the pbuf is ROM/REF, only the ->tot_len and ->len fields are adjusted.
May not be called on a packet queue.
Despite its name, pbuf_realloc cannot grow the size of a pbuf (chain).

Definition at line 430 of file pbuf.c.

5.28.3.15 void pbuf_ref (struct pbuf * p)

Increment the reference count of the pbuf.

Parameters

<i>p</i>	pbuf to increase reference counter of
----------	---------------------------------------

Definition at line 723 of file pbuf.c.

5.28.3.16 `u16_t pbuf_strstr (struct pbuf * p, const char * substr)`

Find occurrence of substr with length substr_len in pbuf p, start at offset start_offset WARNING: in contrast to strstr(), this one does not stop at the first \0 in the pbuf/source string!

Parameters

<i>p</i>	pbuf to search, maximum length is 0xFFFE since 0xFFFF is used as return value 'not found'
<i>substr</i>	string to search for in p, maximum length is 0xFFFE

Returns

0xFFFF if substr was not found in p or the index where it was found

Definition at line 1168 of file pbuf.c.

5.28.3.17 `err_t pbuf_take (struct pbuf * buf, const void * dataptr, u16_t len)`

Copy application supplied data into a pbuf. This function can only be used to copy the equivalent of buf->tot_len data.

Parameters

<i>buf</i>	pbuf to fill with data
<i>dataptr</i>	application supplied data buffer
<i>len</i>	length of the application supplied data buffer

Returns

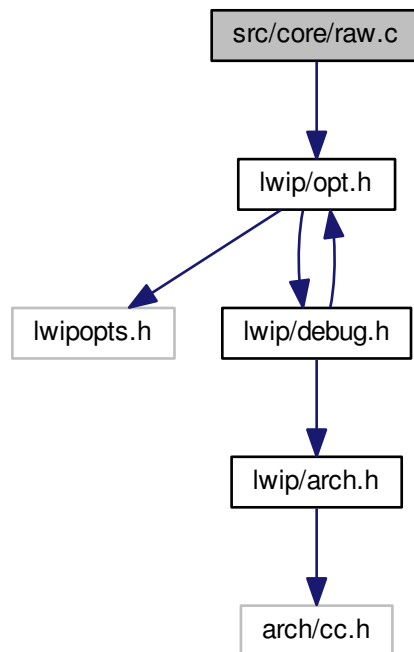
ERR_OK if successful, ERR_MEM if the pbuf is not big enough

Definition at line 966 of file pbuf.c.

5.29 `src/core/raw.c` File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for raw.c:



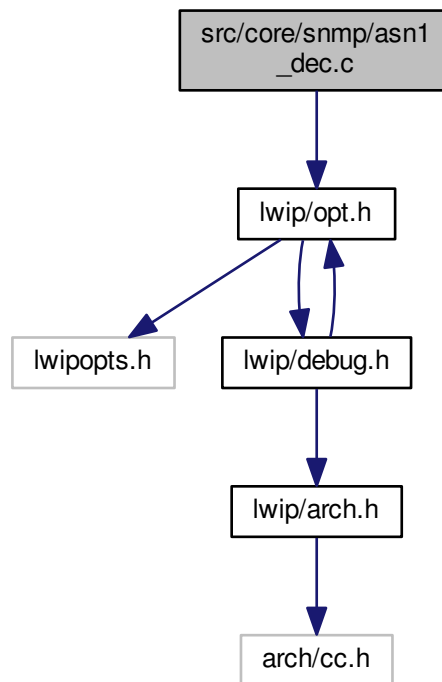
5.29.1 Detailed Description

Implementation of raw protocol PCBs for low-level handling of different types of protocols besides (or overriding) those already available in lwIP.

5.30 src/core/snmp/asn1_dec.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for `asn1_dec.c`:



5.30.1 Detailed Description

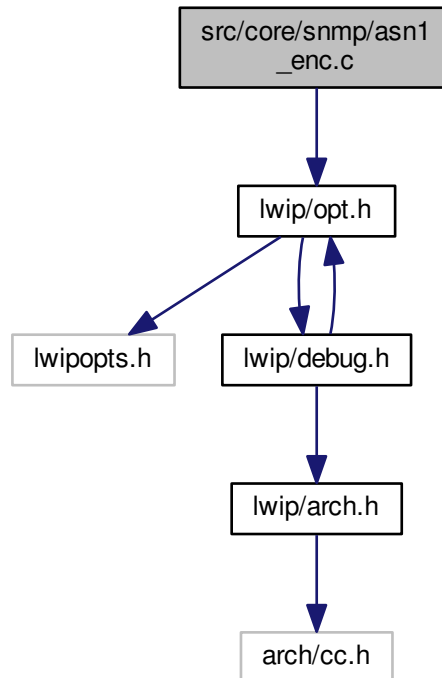
Abstract Syntax Notation One (ISO 8824, 8825) decoding

Todo not optimised (yet), favor correctness over speed, favor speed over size

5.31 src/core/snmp/asn1_enc.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for asn1_enc.c:



5.31.1 Detailed Description

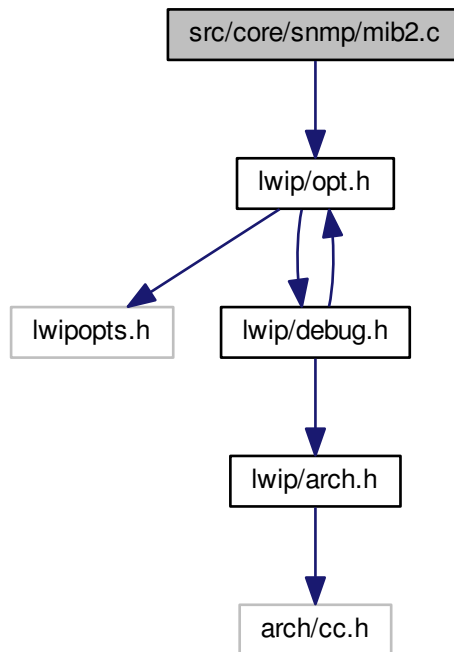
Abstract Syntax Notation One (ISO 8824, 8825) encoding

Todo not optimised (yet), favor correctness over speed, favor speed over size

5.32 src/core/snmp/mib2.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for mib2.c:



5.32.1 Detailed Description

Management Information Base II (RFC1213) objects and functions.

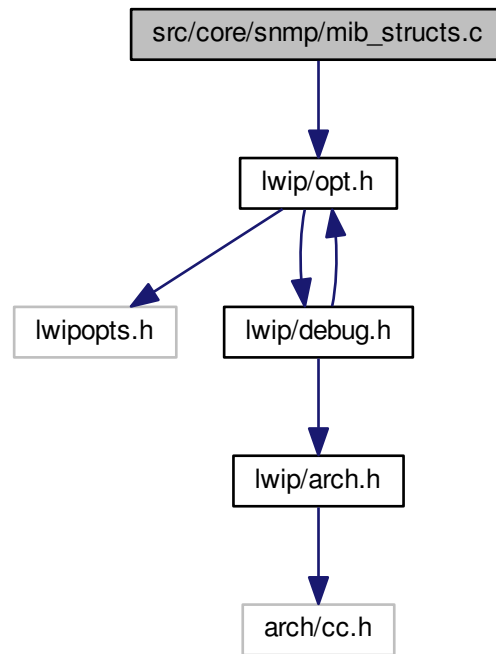
Note

the object identifiers for this MIB-2 and private MIB tree must be kept in sorted ascending order. This to ensure correct getnext operation.

5.33 `src/core/snmp/mib_structs.c` File Reference

```
#include "lwip/opt.h"
```


Include dependency graph for mib_structs.c:



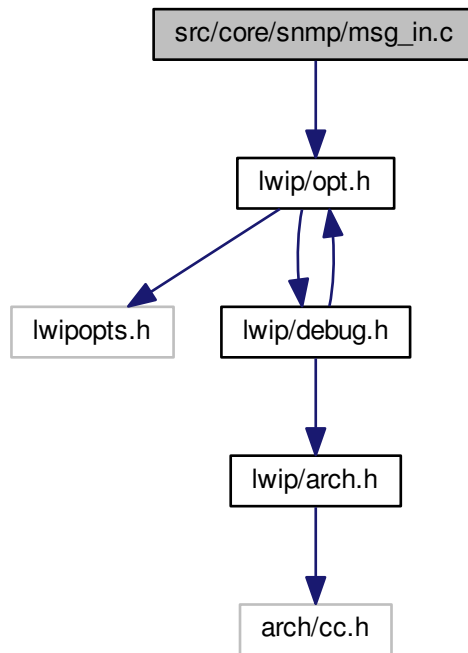
5.33.1 Detailed Description

MIB tree access/construction functions.

5.34 src/core/snmp/msg_in.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for msg_in.c:



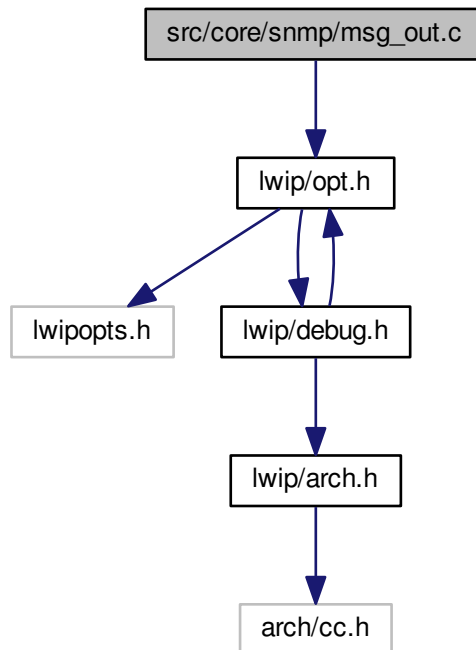
5.34.1 Detailed Description

SNMP input message processing (RFC1157).

5.35 src/core/snmp/msg_out.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for msg_out.c:



5.35.1 Detailed Description

SNMP output message processing (RFC1157).

Output responses and traps are build in two passes:

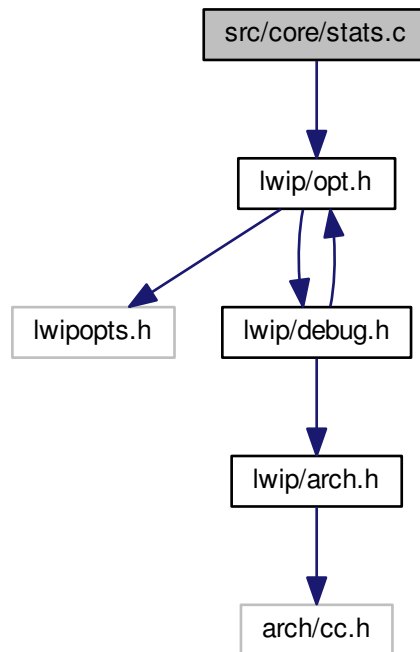
Pass 0: iterate over the output message backwards to determine encoding lengths
Pass 1: the actual forward encoding of internal form into ASN1

The single-pass encoding method described by Comer & Stevens requires extra buffer space and copying for reversal of the packet. The buffer requirement can be prohibitively large for big payloads (≥ 484) therefore we use the two encoding passes.

5.36 src/core/stats.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for stats.c:



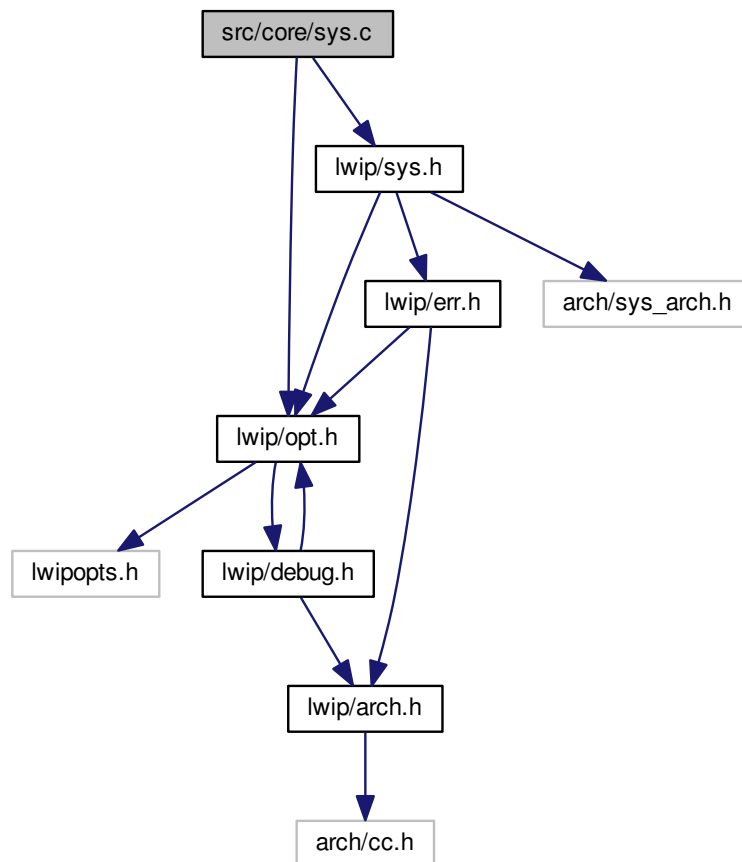
5.36.1 Detailed Description

Statistics module

5.37 `src/core/sys.c` File Reference

```
#include "lwip/opt.h"
#include "lwip/sys.h"
```

Include dependency graph for sys.c:



Functions

- void `sys_msleep` (u32_t ms)

5.37.1 Detailed Description

LwIP Operating System abstraction

5.37.2 Function Documentation

5.37.2.1 void `sys_msleep` (u32_t ms)

Sleep for some ms. Timeouts are NOT processed while sleeping.

Parameters

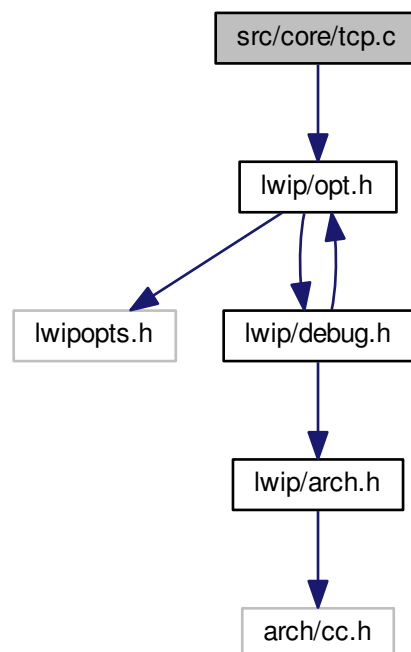
<i>ms</i>	number of milliseconds to sleep
-----------	---------------------------------

Definition at line 55 of file sys.c.

5.38 src/core/tcp.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for tcp.c:



5.38.1 Detailed Description

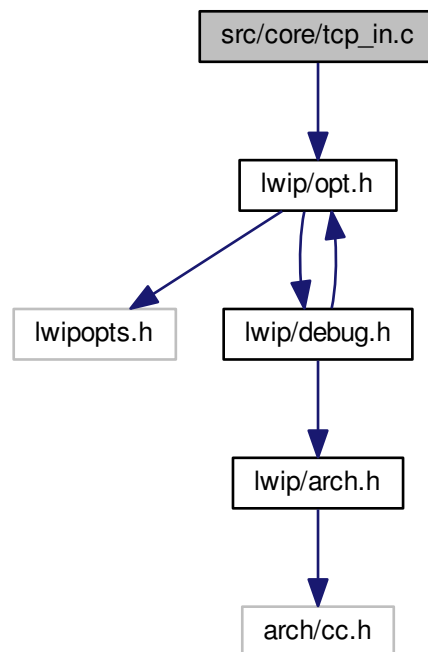
Transmission Control Protocol for IP

This file contains common functions for the TCP implementation, such as functinos for manipulating the data structures and the TCP timer functions. TCP functions related to input and output is found in [tcp_in.c](#) and [tcp_out.c](#) respectively.

5.39 src/core/tcp_in.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for tcp_in.c:



5.39.1 Detailed Description

Transmission Control Protocol, incoming traffic

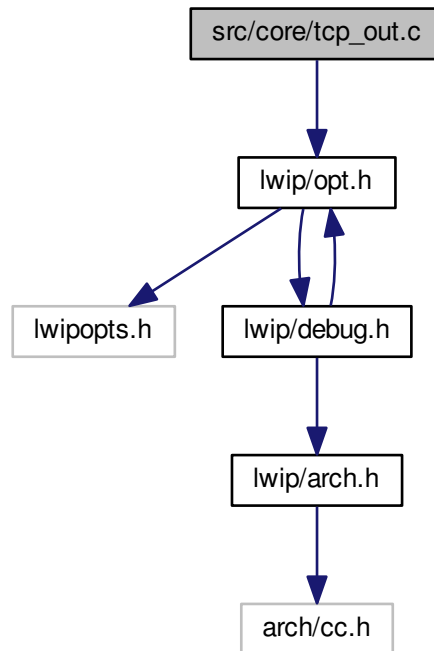
The input processing functions of the TCP layer.

These functions are generally called in the order (`ip_input()` ->) `tcp_input()` -> `*tcp_process()` -> `tcp_receive()` (-> application).

5.40 src/core/tcp_out.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for tcp_out.c:



5.40.1 Detailed Description

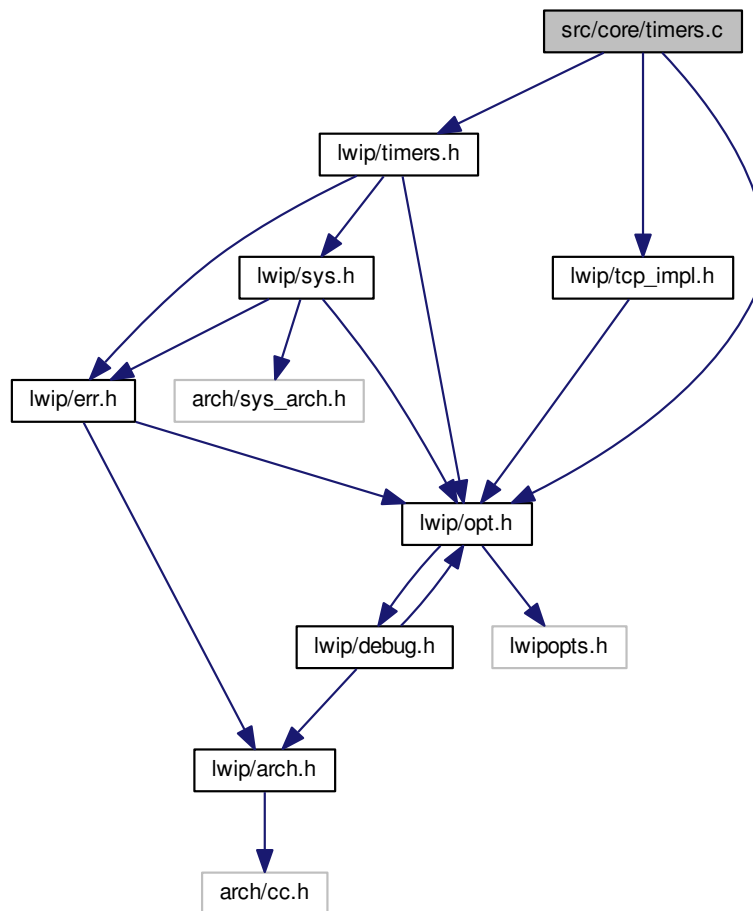
Transmission Control Protocol, outgoing traffic

The output functions of TCP.

5.41 src/core/timers.c File Reference

```
#include "lwip/opt.h"
#include "lwip/timers.h"
#include "lwip/tcp_impl.h"
```


Include dependency graph for timers.c:



Functions

- void [tcp_timer_needed](#) (void)

5.41.1 Detailed Description

Stack-internal timers implementation. This file includes timer callbacks for stack-internal timers as well as functions to set up or stop timers and check for expired timers.

5.41.2 Function Documentation

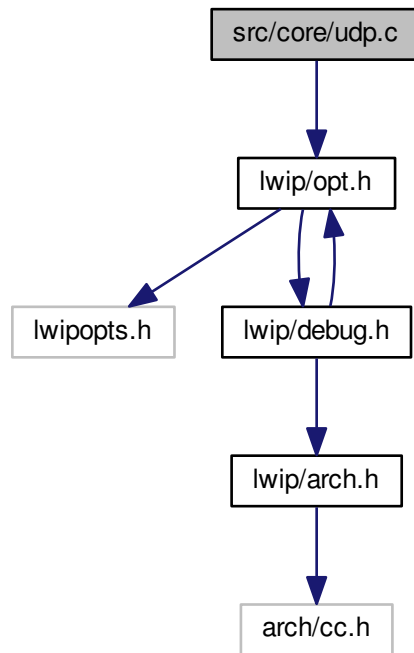
5.41.2.1 void `tcp_timer_needed` (void)

Definition at line 484 of file `timers.c`.

5.42 src/core/udp.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for udp.c:



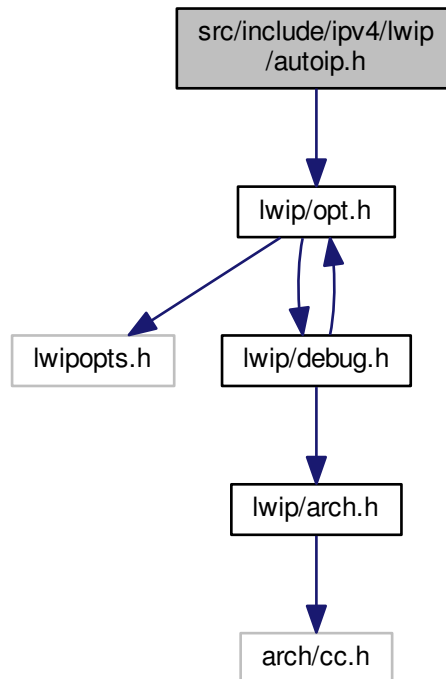
5.42.1 Detailed Description

User Datagram Protocol module

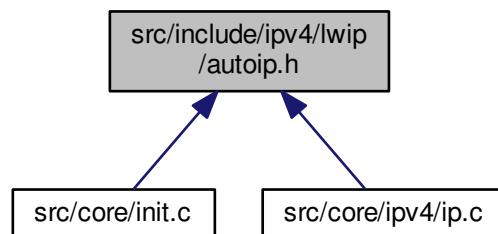
5.43 src/include/ipv4/lwip/autoip.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for autoip.h:



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

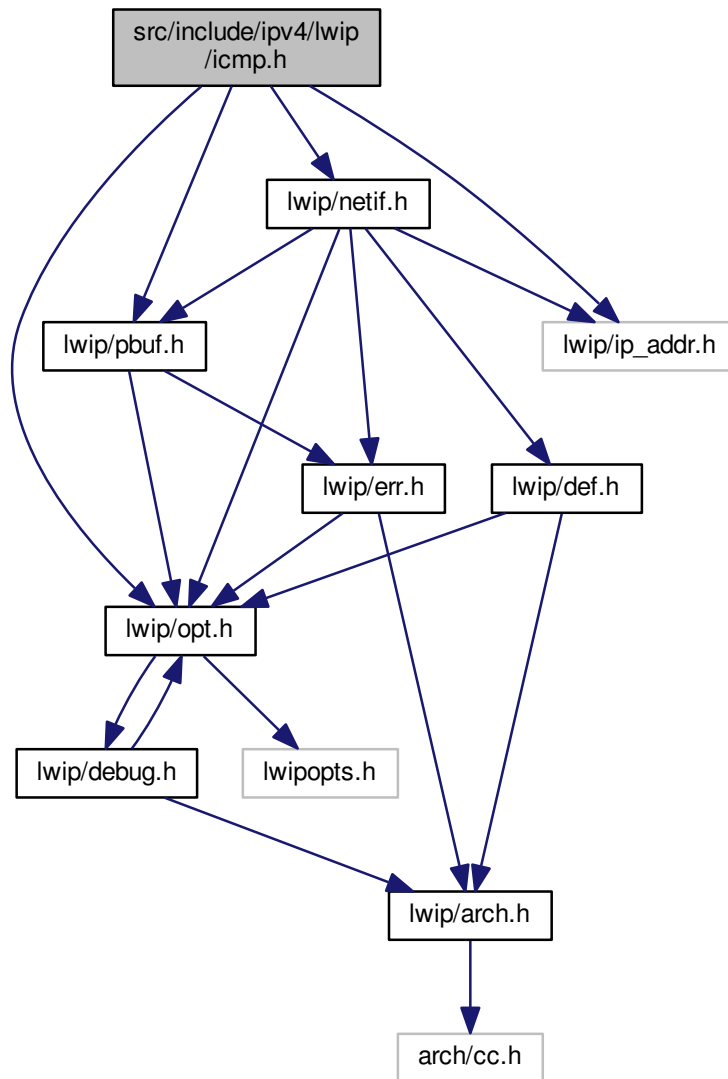


5.43.1 Detailed Description

AutoIP Automatic LinkLocal IP Configuration

5.44 src/include/ipv4/lwip/icmp.h File Reference

```
#include "lwip/opt.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
#include "lwip/netif.h"
Include dependency graph for icmp.h:
```



Data Structures

- struct [icmp_echo_hdr](#)

Macros

- #define [ICMP_ER](#) 0 /* echo reply */

- `#define ICMP_DUR 3 /* destination unreachable */`
- `#define ICMP_SQ 4 /* source quench */`
- `#define ICMP_RD 5 /* redirect */`
- `#define ICMP_ECHO 8 /* echo */`
- `#define ICMP_TE 11 /* time exceeded */`
- `#define ICMP_PP 12 /* parameter problem */`
- `#define ICMP_TS 13 /* timestamp */`
- `#define ICMP_TSR 14 /* timestamp reply */`
- `#define ICMP_IRQ 15 /* information request */`
- `#define ICMP_IR 16 /* information reply */`
- `#define ICMPH_TYPE(hdr) ((hdr)->type)`
- `#define ICMPH_CODE(hdr) ((hdr)->code)`
- `#define ICMPH_TYPE_SET(hdr, t) ((hdr)->type = (t))`
- `#define ICMPH_CODE_SET(hdr, c) ((hdr)->code = (c))`

Enumerations

- `enum icmp_dur_type {
 ICMP_DUR_NET = 0, ICMP_DUR_HOST = 1, ICMP_DUR_PROTO = 2, ICMP_DUR_PORT = 3,
 ICMP_DUR_FRAG = 4, ICMP_DUR_SR = 5 }`
- `enum icmp_te_type { ICMP_TE_TTL = 0, ICMP_TE_FRAG = 1 }`

Variables

- `PACK_STRUCT_BEGIN struct icmp_echo_hdr PACK_STRUCT_STRUCT`

5.44.1 Macro Definition Documentation

5.44.1.1 `#define ICMP_DUR 3 /* destination unreachable */`

Definition at line 45 of file icmp.h.

5.44.1.2 `#define ICMP_ECHO 8 /* echo */`

Definition at line 48 of file icmp.h.

5.44.1.3 `#define ICMP_ER 0 /* echo reply */`

Definition at line 44 of file icmp.h.

5.44.1.4 `#define ICMP_IR 16 /* information reply */`

Definition at line 54 of file icmp.h.

5.44.1.5 `#define ICMP_IRQ 15 /* information request */`

Definition at line 53 of file icmp.h.

5.44.1.6 `#define ICMP_PP 12 /* parameter problem */`

Definition at line 50 of file icmp.h.

5.44.1.7 `#define ICMP_RD 5 /* redirect */`

Definition at line 47 of file icmp.h.

5.44.1.8 `#define ICMP_SQ 4 /* source quench */`

Definition at line 46 of file icmp.h.

5.44.1.9 `#define ICMP_TE 11 /* time exceeded */`

Definition at line 49 of file icmp.h.

5.44.1.10 `#define ICMP_TS 13 /* timestamp */`

Definition at line 51 of file icmp.h.

5.44.1.11 `#define ICMP_TSR 14 /* timestamp reply */`

Definition at line 52 of file icmp.h.

5.44.1.12 `#define ICMPH_CODE(hdr) ((hdr)->code)`

Definition at line 92 of file icmp.h.

5.44.1.13 `#define ICMPH_CODE_SET(hdr, c) ((hdr)->code = (c))`

Definition at line 96 of file icmp.h.

5.44.1.14 `#define ICMPH_TYPE(hdr) ((hdr)->type)`

Definition at line 91 of file icmp.h.

5.44.1.15 `#define ICMPH_TYPE_SET(hdr, t) ((hdr)->type = (t))`

Combines type and code to an u16_t

Definition at line 95 of file icmp.h.

5.44.2 Enumeration Type Documentation

5.44.2.1 `enum icmp_dur_type`

Enumerator

ICMP_DUR_NET
ICMP_DUR_HOST
ICMP_DUR_PROTO
ICMP_DUR_PORT
ICMP_DUR_FRAG
ICMP_DUR_SR

Definition at line 56 of file icmp.h.

5.44.2.2 enum icmp_te_type

Enumerator

*ICMP_TE_TTL**ICMP_TE_FRAG*

Definition at line 65 of file icmp.h.

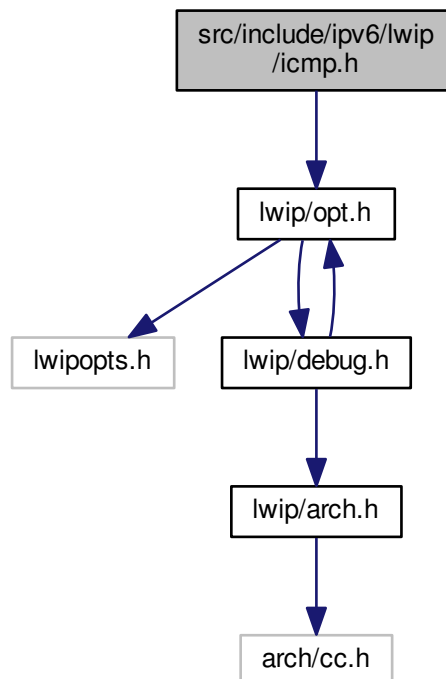
5.44.3 Variable Documentation

5.44.3.1 PACK_STRUCT_END PACK_STRUCT_BEGIN struct ip_addr2 PACK_STRUCT_STRUCT

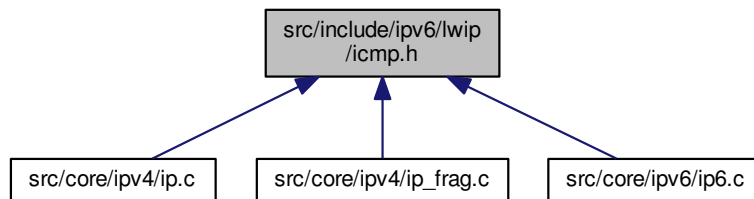
5.45 src/include/ipv6/lwip/icmp.h File Reference

#include "lwip/opt.h"

Include dependency graph for icmp.h:



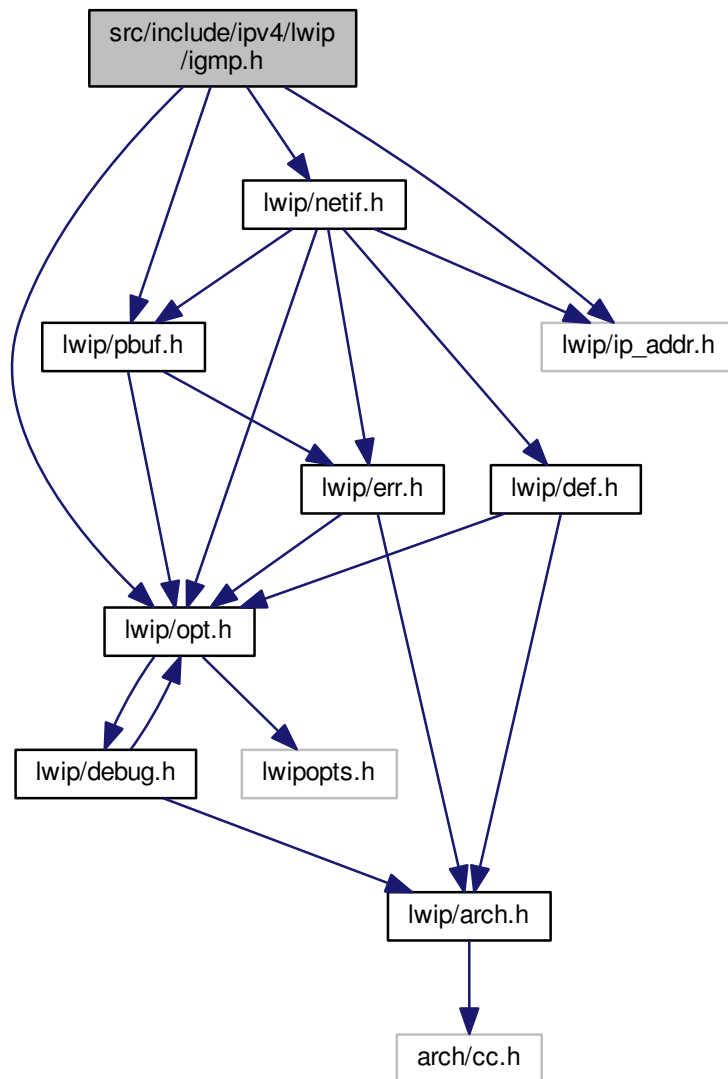
This graph shows which files directly or indirectly include this file:



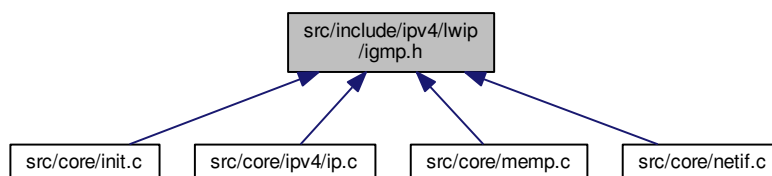
5.46 `src/include/ipv4/lwip/igmp.h` File Reference

```
#include "lwip/opt.h"
#include "lwip/ip_addr.h"
#include "lwip/netif.h"
#include "lwip/pbuf.h"
```


Include dependency graph for igmp.h:

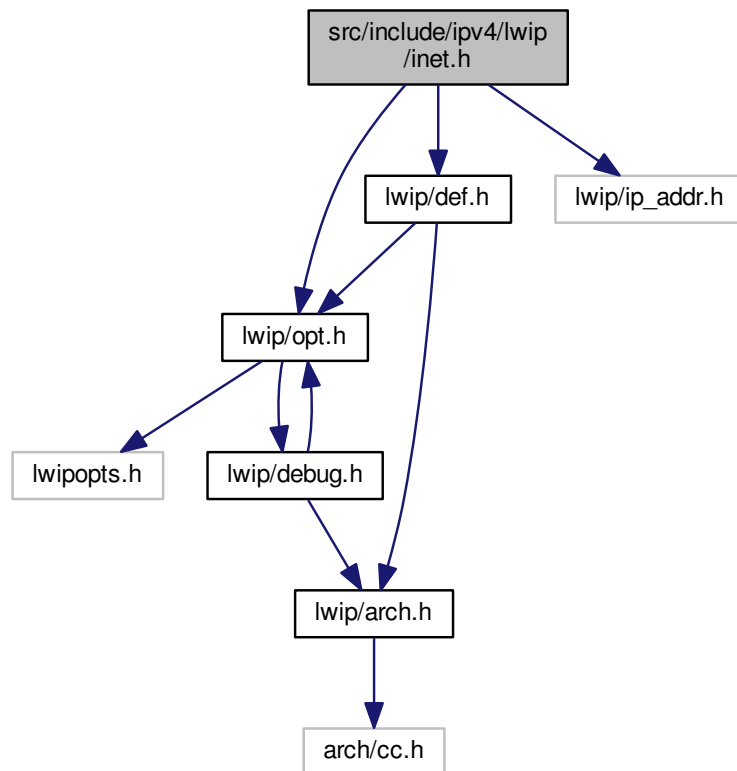


This graph shows which files directly or indirectly include this file:



5.47 src/include/ipv4/lwip/inet.h File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/ip_addr.h"
Include dependency graph for inet.h:
```



Data Structures

- struct [in_addr](#)

Macros

- `#define INADDR_NONE IPADDR_NONE`
- `#define INADDR_LOOPBACK IPADDR_LOOPBACK`
- `#define INADDR_ANY IPADDR_ANY`
- `#define INADDR_BROADCAST IPADDR_BROADCAST`
- `#define IN_CLASSA(a) IP_CLASSA(a)`
- `#define IN_CLASSA_NET IP_CLASSA_NET`
- `#define IN_CLASSA_NSHIFT IP_CLASSA_NSHIFT`
- `#define IN_CLASSA_HOST IP_CLASSA_HOST`
- `#define IN_CLASSA_MAX IP_CLASSA_MAX`
- `#define IN_CLASSB(b) IP_CLASSB(b)`

- `#define IN_CLASSB_NET IP_CLASSB_NET`
- `#define IN_CLASSB_NSHIFT IP_CLASSB_NSHIFT`
- `#define IN_CLASSB_HOST IP_CLASSB_HOST`
- `#define IN_CLASSB_MAX IP_CLASSB_MAX`
- `#define IN_CLASSC(c) IP_CLASSC(c)`
- `#define IN_CLASSC_NET IP_CLASSC_NET`
- `#define IN_CLASSC_NSHIFT IP_CLASSC_NSHIFT`
- `#define IN_CLASSC_HOST IP_CLASSC_HOST`
- `#define IN_CLASSC_MAX IP_CLASSC_MAX`
- `#define IN_CLASSD(d) IP_CLASSD(d)`
- `#define IN_CLASSD_NET IP_CLASSD_NET` /* These ones aren't really */
- `#define IN_CLASSD_NSHIFT IP_CLASSD_NSHIFT` /* net and host fields, but */
- `#define IN_CLASSD_HOST IP_CLASSD_HOST` /* routing needn't know. */
- `#define IN_CLASSD_MAX IP_CLASSD_MAX`
- `#define IN_MULTICAST(a) IP_MULTICAST(a)`
- `#define IN_EXPERIMENTAL(a) IP_EXPERIMENTAL(a)`
- `#define IN_BADCLASS(a) IP_BADCLASS(a)`
- `#define IN_LOOPBACKNET IP_LOOPBACKNET`
- `#define inet_addr_from_ipaddr(target_inaddr, source_ipaddr) ((target_inaddr)->s_addr = ip4_addr_get_u32(source_ipaddr))`
- `#define inet_addr_to_ipaddr(target_ipaddr, source_inaddr) (ip4_addr_set_u32(target_ipaddr, (source_inaddr)->s_addr))`
- `#define inet_addr_to_ipaddr_p(target_ipaddr_p, source_inaddr) ((target_ipaddr_p) = (ip_addr_t*)&((source_inaddr)->s_addr))`
- `#define inet_addr(cp) ipaddr_addr(cp)`
- `#define inet_aton(cp, addr) ipaddr_aton(cp, (ip_addr_t*)addr)`
- `#define inet_ntoa(addr) ipaddr_ntoa((ip_addr_t*)&(addr))`
- `#define inet_ntoa_r(addr, buf, buflen) ipaddr_ntoa_r((ip_addr_t*)&(addr), buf, buflen)`

5.47.1 Macro Definition Documentation

5.47.1.1 `#define IN_BADCLASS(a) IP_BADCLASS(a)`

Definition at line 88 of file inet.h.

5.47.1.2 `#define IN_CLASSA(a) IP_CLASSA(a)`

Definition at line 61 of file inet.h.

5.47.1.3 `#define IN_CLASSA_HOST IP_CLASSA_HOST`

Definition at line 64 of file inet.h.

5.47.1.4 `#define IN_CLASSA_MAX IP_CLASSA_MAX`

Definition at line 65 of file inet.h.

5.47.1.5 `#define IN_CLASSA_NET IP_CLASSA_NET`

Definition at line 62 of file inet.h.

5.47.1.6 `#define IN_CLASSA_NSHIFT IP_CLASSA_NSHIFT`

Definition at line 63 of file inet.h.

5.47.1.7 `#define IN_CLASSB(b) IP_CLASSB(b)`

Definition at line 67 of file inet.h.

5.47.1.8 `#define IN_CLASSB_HOST IP_CLASSB_HOST`

Definition at line 70 of file inet.h.

5.47.1.9 `#define IN_CLASSB_MAX IP_CLASSB_MAX`

Definition at line 71 of file inet.h.

5.47.1.10 `#define IN_CLASSB_NET IP_CLASSB_NET`

Definition at line 68 of file inet.h.

5.47.1.11 `#define IN_CLASSB_NSHIFT IP_CLASSB_NSHIFT`

Definition at line 69 of file inet.h.

5.47.1.12 `#define IN_CLASSC(c) IP_CLASSC(c)`

Definition at line 73 of file inet.h.

5.47.1.13 `#define IN_CLASSC_HOST IP_CLASSC_HOST`

Definition at line 76 of file inet.h.

5.47.1.14 `#define IN_CLASSC_MAX IP_CLASSC_MAX`

Definition at line 77 of file inet.h.

5.47.1.15 `#define IN_CLASSC_NET IP_CLASSC_NET`

Definition at line 74 of file inet.h.

5.47.1.16 `#define IN_CLASSC_NSHIFT IP_CLASSC_NSHIFT`

Definition at line 75 of file inet.h.

5.47.1.17 `#define IN_CLASSD(d) IP_CLASSD(d)`

Definition at line 79 of file inet.h.

5.47.1.18 **#define IN_CLASSD_HOST IP_CLASSD_HOST** /* routing needn't know. */

Definition at line 82 of file inet.h.

5.47.1.19 **#define IN_CLASSD_MAX IP_CLASSD_MAX**

Definition at line 83 of file inet.h.

5.47.1.20 **#define IN_CLASSD_NET IP_CLASSD_NET** /* These ones aren't really */

Definition at line 80 of file inet.h.

5.47.1.21 **#define IN_CLASSD_NSHIFT IP_CLASSD_NSHIFT** /* net and host fields, but */

Definition at line 81 of file inet.h.

5.47.1.22 **#define IN_EXPERIMENTAL(a) IP_EXPERIMENTAL(a)**

Definition at line 87 of file inet.h.

5.47.1.23 **#define IN_LOOPBACKNET IP_LOOPBACKNET**

Definition at line 90 of file inet.h.

5.47.1.24 **#define IN_MULTICAST(a) IP_MULTICAST(a)**

Definition at line 85 of file inet.h.

5.47.1.25 **#define INADDR_ANY IPADDR_ANY**

0.0.0.0

Definition at line 53 of file inet.h.

5.47.1.26 **#define INADDR_BROADCAST IPADDR_BROADCAST**

255.255.255.255

Definition at line 55 of file inet.h.

5.47.1.27 **#define INADDR_LOOPBACK IPADDR_LOOPBACK**

127.0.0.1

Definition at line 51 of file inet.h.

5.47.1.28 **#define INADDR_NONE IPADDR_NONE**

255.255.255.255

Definition at line 49 of file inet.h.

5.47.1.29 `#define inet_addr(cp) ipaddr_addr(cp)`

Definition at line 98 of file inet.h.

5.47.1.30 `#define inet_addr_from_ipaddr(target_inaddr, source_ipaddr) ((target_inaddr)->s_addr = ip4_addr_get_u32(source_ipaddr))`

Definition at line 92 of file inet.h.

5.47.1.31 `#define inet_addr_to_ipaddr(target_ipaddr, source_inaddr) (ip4_addr_set_u32(target_ipaddr, (source_inaddr)->s_addr))`

Definition at line 93 of file inet.h.

5.47.1.32 `#define inet_addr_to_ipaddr_p(target_ipaddr_p, source_inaddr) ((target_ipaddr_p) = (ip_addr_t*)&((source_inaddr)->s_addr))`

Definition at line 95 of file inet.h.

5.47.1.33 `#define inet_aton(cp, addr) ipaddr_aton(cp, (ip_addr_t*)addr)`

Definition at line 99 of file inet.h.

5.47.1.34 `#define inet_ntoa(addr) ipaddr_ntoa((ip_addr_t*)&(addr))`

Definition at line 100 of file inet.h.

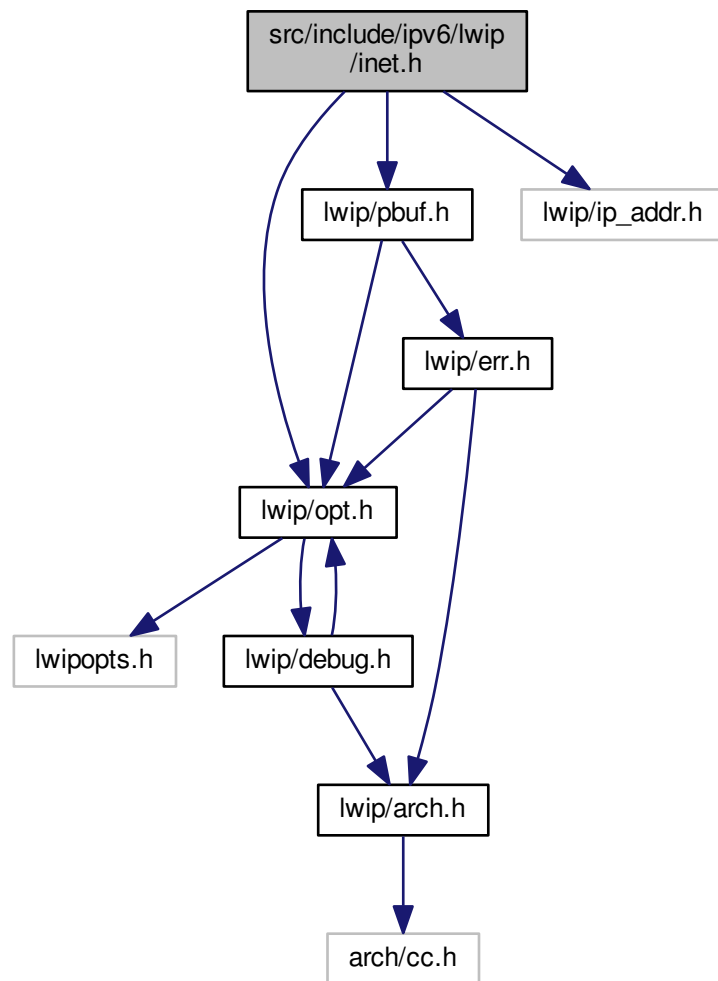
5.47.1.35 `#define inet_ntoa_r(addr, buf, buflen) ipaddr_ntoa_r((ip_addr_t*)&(addr), buf, buflen)`

Definition at line 101 of file inet.h.

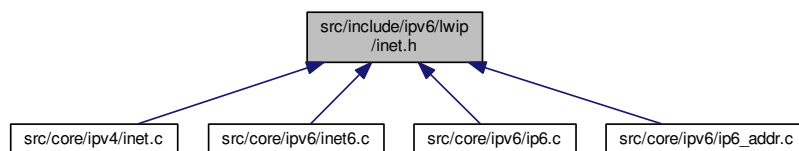
5.48 src/include/ipv6/lwip/inet.h File Reference

```
#include "lwip/opt.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
```

Include dependency graph for inet.h:



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



Functions

- `u16_t inet_chksum (void *data, u16_t len)`

- `u16_t inet_chksum_pbuf` (struct `pbuf` *`p`)
- `u16_t inet_chksum_pseudo` (struct `pbuf` *`p`, struct `ip_addr` *`src`, struct `ip_addr` *`dest`, `u8_t` `proto`, `u32_t` `proto_len`)
- `u32_t inet_addr` (const char *`cp`)
- `s8_t inet_aton` (const char *`cp`, struct `in_addr` *`addr`)
- `u16_t htons` (`u16_t` `n`)
- `u16_t ntohs` (`u16_t` `n`)
- `u32_t htonl` (`u32_t` `n`)
- `u32_t ntohl` (`u32_t` `n`)

5.48.1 Function Documentation

5.48.1.1 `u32_t htonl (u32_t n)`

5.48.1.2 `u16_t htons (u16_t n)`

5.48.1.3 `u32_t inet_addr (const char * cp)`

5.48.1.4 `s8_t inet_aton (const char * cp, struct in_addr * addr)`

5.48.1.5 `u16_t inet_chksum (void * data, u16_t len)`

Definition at line 396 of file `inet_chksum.c`.

5.48.1.6 `u16_t inet_chksum_pbuf (struct pbuf * p)`

Calculate a checksum over a chain of pbufs (without pseudo-header, much like `inet_chksum` only pbufs are used).

Parameters

<code>p</code>	pbuf chain over that the checksum should be calculated
----------------	--

Returns

checksum (as `u16_t`) to be saved directly in the protocol header

Definition at line 409 of file `inet_chksum.c`.

5.48.1.7 `u16_t inet_chksum_pseudo (struct pbuf * p, struct ip_addr * src, struct ip_addr * dest, u8_t proto, u32_t proto_len)`

Definition at line 80 of file `inet6.c`.

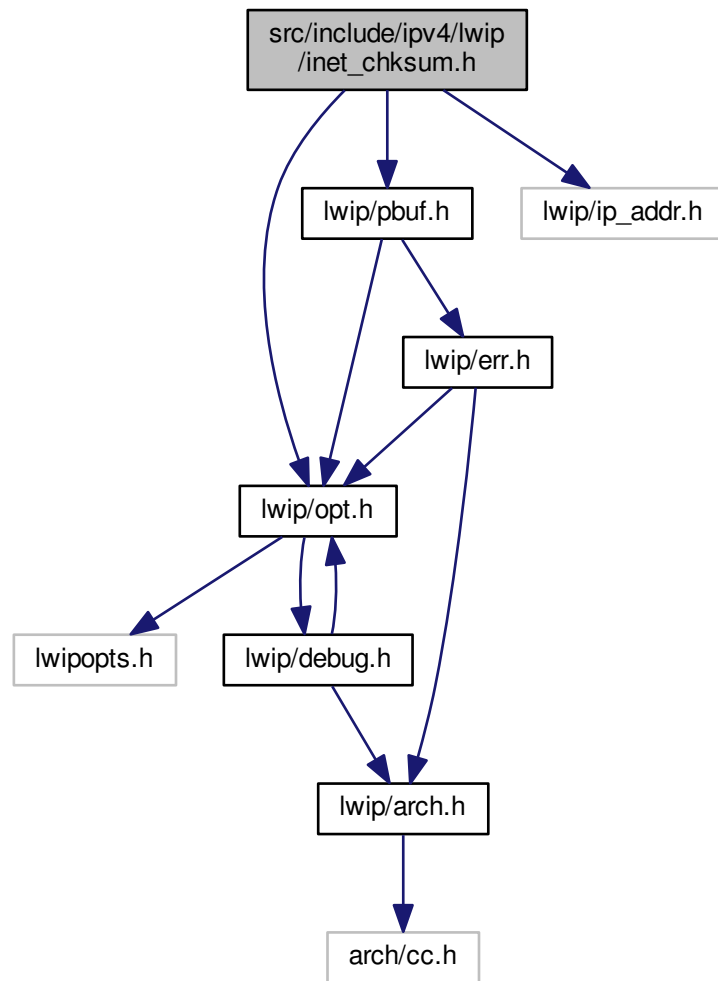
5.48.1.8 `u32_t ntohl (u32_t n)`

5.48.1.9 `u16_t ntohs (u16_t n)`

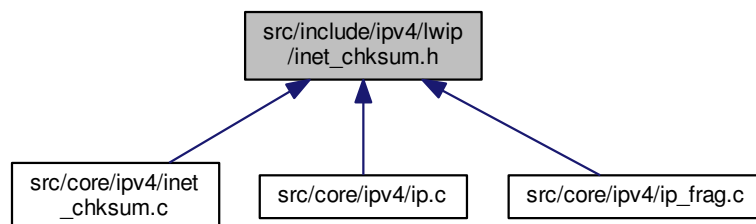
5.49 src/include/ipv4/lwip/inet_chksum.h File Reference

```
#include "lwip/opt.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
```


Include dependency graph for inet_chksum.h:



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



Macros

- `#define SWAP_BYTES_IN_WORD(w) (((w) & 0xff) << 8) | (((w) & 0xff00) >> 8)`
- `#define FOLD_U32T(u) (((u) >> 16) + ((u) & 0x0000ffffUL))`
- `#define LWIP_CHKSUM_COPY_ALGORITHM 0`

Functions

- `u16_t inet_chksum (void *dataptr, u16_t len)`
- `u16_t inet_chksum_pbuf (struct pbuf *p)`
- `u16_t inet_chksum_pseudo (struct pbuf *p, ip_addr_t *src, ip_addr_t *dest, u8_t proto, u16_t proto_len)`
- `u16_t inet_chksum_pseudo_partial (struct pbuf *p, ip_addr_t *src, ip_addr_t *dest, u8_t proto, u16_t proto_len, u16_t chksum_len)`

5.49.1 Macro Definition Documentation

5.49.1.1 `#define FOLD_U32T(u) (((u) >> 16) + ((u) & 0x0000ffffUL))`

Split an `u32_t` in two `u16_ts` and add them up

Definition at line 53 of file `inet_chksum.h`.

5.49.1.2 `#define LWIP_CHKSUM_COPY_ALGORITHM 0`

Definition at line 66 of file `inet_chksum.h`.

5.49.1.3 `#define SWAP_BYTES_IN_WORD(w) (((w) & 0xff) << 8) | (((w) & 0xff00) >> 8)`

Swap the bytes in an `u16_t`: much like `htons()` for little-endian

Definition at line 47 of file `inet_chksum.h`.

5.49.2 Function Documentation

5.49.2.1 `u16_t inet_chksum (void * dataptr, u16_t len)`

Definition at line 396 of file `inet_chksum.c`.

5.49.2.2 `u16_t inet_chksum_pbuf (struct pbuf * p)`

Calculate a checksum over a chain of pbufs (without pseudo-header, much like `inet_chksum` only pbufs are used).

Parameters

<i>p</i>	pbuf chain over that the checksum should be calculated
----------	--

Returns

checksum (as `u16_t`) to be saved directly in the protocol header

Definition at line 409 of file `inet_chksum.c`.

5.49.2.3 `u16_t inet_chksum_pseudo (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t proto, u16_t proto_len)`

Definition at line 272 of file `inet_chksum.c`.

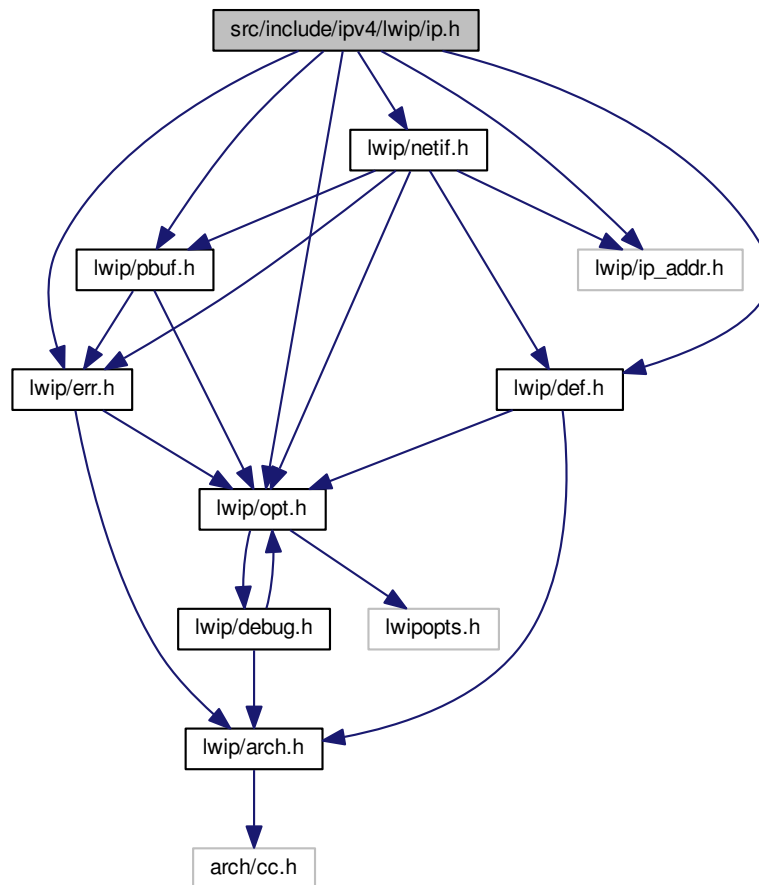
5.49.2.4 `u16_t inet_chksum_pseudo_partial (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t proto, u16_t proto_len, u16_t chksum_len)`

Definition at line 332 of file `inet_chksum.c`.

5.50 src/include/ipv4/lwip/ip.h File Reference

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
#include "lwip/err.h"
#include "lwip/netif.h"
```

Include dependency graph for `ip.h`:



Data Structures

- struct [ip_pcb](#)
- struct [ip_hdr](#)

Macros

- `#define IP_OPTIONS_SEND LWIP_IGMP`
- `#define IP_HLEN 20`
- `#define IP_PROTO_ICMP 1`
- `#define IP_PROTO_IGMP 2`
- `#define IP_PROTO_UDP 17`
- `#define IP_PROTO_UDPLITE 136`
- `#define IP_PROTO_TCP 6`
- `#define IP_HDRINCL NULL`
- `#define IP_PCB_ADDRHINT`
- `#define IP_PCB`
- `#define SOF_ACCEPTCONN 0x02U /* socket has had listen() */`
- `#define SOF_REUSEADDR 0x04U /* allow local address reuse */`
- `#define SOF_KEEPAIVE 0x08U /* keep connections alive */`
- `#define SOF_BROADCAST 0x20U /* permit to send and to receive broadcast messages (see IP_SOF_BROADCAST option) */`
- `#define SOF_LINGER 0x80U /* linger on close if data present */`
- `#define SOF_INHERITED (SOF_REUSEADDR|SOF_KEEPAIVE|SOF_LINGER/*|SOF_DEBUG|SOF_DONTROUTE|SOF_OOBLINE*/)`
- `#define IP_RF 0x8000U /* reserved fragment flag */`
- `#define IP_DF 0x4000U /* dont fragment flag */`
- `#define IP_MF 0x2000U /* more fragments flag */`
- `#define IP_OFFMASK 0x1fffU /* mask for fragmenting bits */`
- `#define IPH_V(hdr) ((hdr)->_v_hl >> 4)`
- `#define IPH_HL(hdr) ((hdr)->_v_hl & 0x0f)`
- `#define IPH_TOS(hdr) ((hdr)->_tos)`
- `#define IPH_LEN(hdr) ((hdr)->_len)`
- `#define IPH_ID(hdr) ((hdr)->_id)`
- `#define IPH_OFFSET(hdr) ((hdr)->_offset)`
- `#define IPH_TTL(hdr) ((hdr)->_ttl)`
- `#define IPH_PROTO(hdr) ((hdr)->_proto)`
- `#define IPH_CHKSUM(hdr) ((hdr)->_checksum)`
- `#define IPH_VHL_SET(hdr, v, hl) (hdr)->_v_hl = (((v) << 4) | (hl))`
- `#define IPH_TOS_SET(hdr, tos) (hdr)->_tos = (tos)`
- `#define IPH_LEN_SET(hdr, len) (hdr)->_len = (len)`
- `#define IPH_ID_SET(hdr, id) (hdr)->_id = (id)`
- `#define IPH_OFFSET_SET(hdr, off) (hdr)->_offset = (off)`
- `#define IPH_TTL_SET(hdr, ttl) (hdr)->_ttl = (u8_t)(ttl)`
- `#define IPH_PROTO_SET(hdr, proto) (hdr)->_proto = (u8_t)(proto)`
- `#define IPH_CHKSUM_SET(hdr, checksum) (hdr)->_checksum = (checksum)`
- `#define ip_init() /* Compatibility define, not init needed. */`
- `#define ip_current_netif() (current_netif)`
- `#define ip_current_header() (current_header)`
- `#define ip_current_src_addr() (¤t_iphdr_src)`
- `#define ip_current_dest_addr() (¤t_iphdr_dest)`
- `#define ip_get_option(pcb, opt) ((pcb)->so_options & (opt))`
- `#define ip_set_option(pcb, opt) ((pcb)->so_options |= (opt))`
- `#define ip_reset_option(pcb, opt) ((pcb)->so_options &= ~(opt))`
- `#define ip_debug_print(p)`

Functions

- struct `netif` * `ip_route` (`ip_addr_t` *dest)
- `err_t` `ip_input` (struct `pbuf` *p, struct `netif` *inp)
- `err_t` `ip_output` (struct `pbuf` *p, `ip_addr_t` *src, `ip_addr_t` *dest, `u8_t` ttl, `u8_t` tos, `u8_t` proto)
- `err_t` `ip_output_if` (struct `pbuf` *p, `ip_addr_t` *src, `ip_addr_t` *dest, `u8_t` ttl, `u8_t` tos, `u8_t` proto, struct `netif` *netif)

Variables

- `PACK_STRUCT_BEGIN` struct `ip_hdr` `PACK_STRUCT_STRUCT`
- struct `netif` * `current_netif`
- const struct `ip_hdr` * `current_header`
- `ip_addr_t` `current_iphdr_src`
- `ip_addr_t` `current_iphdr_dest`

5.50.1 Macro Definition Documentation

5.50.1.1 #define `ip_current_dest_addr()` (¤t_iphdr_dest)

Destination IP address of `current_header`

Definition at line 202 of file ip.h.

5.50.1.2 #define `ip_current_header()` (current_header)

Get the IP header of the current packet. This function must only be called from a receive callback (`udp_rcv`, `raw_rcv`, `tcp_accept`). It will return NULL otherwise.

Definition at line 198 of file ip.h.

5.50.1.3 #define `ip_current_netif()` (current_netif)

Get the interface that received the current packet. This function must only be called from a receive callback (`udp_rcv`, `raw_rcv`, `tcp_accept`). It will return NULL otherwise.

Definition at line 194 of file ip.h.

5.50.1.4 #define `ip_current_src_addr()` (¤t_iphdr_src)

Source IP address of `current_header`

Definition at line 200 of file ip.h.

5.50.1.5 #define `ip_debug_print(p)`

Definition at line 214 of file ip.h.

5.50.1.6 #define `IP_DF` 0x4000U /* dont fragment flag */

Definition at line 128 of file ip.h.

5.50.1.7 `#define ip_get_option(pcb, opt) ((pcb)->so_options & (opt))`

Gets an IP pcb option (SOF_* flags)

Definition at line 205 of file ip.h.

5.50.1.8 `#define IP_HDRINCL NULL`

Definition at line 64 of file ip.h.

5.50.1.9 `#define IP_HLEN 20`

Definition at line 50 of file ip.h.

5.50.1.10 `#define ip_init(void) /* Compatibility define, not init needed. */`

Definition at line 174 of file ip.h.

5.50.1.11 `#define IP_MF 0x2000U /* more fragments flag */`

Definition at line 129 of file ip.h.

5.50.1.12 `#define IP_OFFMASK 0x1fffU /* mask for fragmenting bits */`

Definition at line 130 of file ip.h.

5.50.1.13 `#define IP_OPTIONS_SEND LWIP_IGMP`

Currently, the function `ip_output_if_opt()` is only used with IGMP

Definition at line 48 of file ip.h.

5.50.1.14 `#define IP_PCB`

Value:

```
/* ip addresses in network byte order */ \
ip_addr_t local_ip; \
ip_addr_t remote_ip; \
/* Socket options */ \
u8_t so_options; \
/* Type Of Service */ \
u8_t tos; \
/* Time To Live */ \
u8_t ttl \
/* link layer address resolution hint */ \
IP_PCB_ADDRHINT
```

Definition at line 76 of file ip.h.

5.50.1.15 `#define IP_PCB_ADDRHINT`

Definition at line 69 of file ip.h.

5.50.1.16 `#define IP_PROTO_ICMP 1`

Definition at line 52 of file ip.h.

5.50.1.17 `#define IP_PROTO_IGMP 2`

Definition at line 53 of file ip.h.

5.50.1.18 `#define IP_PROTO_TCP 6`

Definition at line 56 of file ip.h.

5.50.1.19 `#define IP_PROTO_UDP 17`

Definition at line 54 of file ip.h.

5.50.1.20 `#define IP_PROTO_UDPLITE 136`

Definition at line 55 of file ip.h.

5.50.1.21 `#define ip_reset_option(pcb, opt) ((pcb)->so_options &= ~(opt))`

Resets an IP pcb option (SOF_* flags)

Definition at line 209 of file ip.h.

5.50.1.22 `#define IP_RF 0x8000U /* reserved fragment flag */`

Definition at line 127 of file ip.h.

5.50.1.23 `#define ip_set_option(pcb, opt) ((pcb)->so_options |= (opt))`

Sets an IP pcb option (SOF_* flags)

Definition at line 207 of file ip.h.

5.50.1.24 `#define IPH_CHKSUM(hdr) ((hdr)->_chksum)`

Definition at line 154 of file ip.h.

5.50.1.25 `#define IPH_CHKSUM_SET(hdr, chksum) (hdr)->_chksum = (chksum)`

Definition at line 163 of file ip.h.

5.50.1.26 `#define IPH_HL(hdr) ((hdr)->_v_hl & 0x0f)`

Definition at line 147 of file ip.h.

5.50.1.27 `#define IPH_ID(hdr) ((hdr)->_id)`

Definition at line 150 of file ip.h.

5.50.1.28 `#define IPH_ID_SET(hdr, id) ((hdr)->_id = (id))`

Definition at line 159 of file ip.h.

5.50.1.29 `#define IPH_LEN(hdr) ((hdr)->_len)`

Definition at line 149 of file ip.h.

5.50.1.30 `#define IPH_LEN_SET(hdr, len) ((hdr)->_len = (len))`

Definition at line 158 of file ip.h.

5.50.1.31 `#define IPH_OFFSET(hdr) ((hdr)->_offset)`

Definition at line 151 of file ip.h.

5.50.1.32 `#define IPH_OFFSET_SET(hdr, off) ((hdr)->_offset = (off))`

Definition at line 160 of file ip.h.

5.50.1.33 `#define IPH_PROTO(hdr) ((hdr)->_proto)`

Definition at line 153 of file ip.h.

5.50.1.34 `#define IPH_PROTO_SET(hdr, proto) ((hdr)->_proto = (u8_t)(proto))`

Definition at line 162 of file ip.h.

5.50.1.35 `#define IPH_TOS(hdr) ((hdr)->_tos)`

Definition at line 148 of file ip.h.

5.50.1.36 `#define IPH_TOS_SET(hdr, tos) ((hdr)->_tos = (tos))`

Definition at line 157 of file ip.h.

5.50.1.37 `#define IPH_TTL(hdr) ((hdr)->_ttl)`

Definition at line 152 of file ip.h.

5.50.1.38 `#define IPH_TTL_SET(hdr, ttl) ((hdr)->_ttl = (u8_t)(ttl))`

Definition at line 161 of file ip.h.

5.50.1.39 `#define IPH_V(hdr) ((hdr)->_v_hl >> 4)`

Definition at line 146 of file ip.h.

5.50.1.40 `#define IPH_VHL_SET(hdr, v, hl) (hdr)->_v_hl = (((v) << 4) | (hl))`

Definition at line 156 of file ip.h.

5.50.1.41 `#define SOF_ACCEPTCONN 0x02U /* socket has had listen() */`

Definition at line 98 of file ip.h.

5.50.1.42 `#define SOF_BROADCAST 0x20U /* permit to send and to receive broadcast messages (see IP_SOF_BROADCAST option) */`

Definition at line 102 of file ip.h.

5.50.1.43 `#define SOF_INHERITED (SOF_REUSEADDR|SOF_KEEPALIVE|SOF_LINGER/*|SOF_DEBUG|SOF_DON←
TROUTE|SOF_OOBLINE*/)`

Definition at line 109 of file ip.h.

5.50.1.44 `#define SOF_KEEPALIVE 0x08U /* keep connections alive */`

Definition at line 100 of file ip.h.

5.50.1.45 `#define SOF_LINGER 0x80U /* linger on close if data present */`

Definition at line 104 of file ip.h.

5.50.1.46 `#define SOF_REUSEADDR 0x04U /* allow local address reuse */`

Definition at line 99 of file ip.h.

5.50.2 Function Documentation

5.50.2.1 `err_t ip_input (struct pbuf * p, struct netif * inp)`

This function is called by the network interface device driver when an IP packet is received. The function does the basic checks of the IP header such as packet size being at least larger than the header size etc. If the packet was not destined for us, the packet is forwarded (using ip_forward). The IP checksum is always checked.

Finally, the packet is sent to the upper layer protocol input function.

Parameters

<i>p</i>	the received IP packet (p->payload points to IP header)
<i>inp</i>	the netif on which this packet was received

Returns

ERR_OK if the packet was processed (could return ERR_* if it wasn't processed, but currently always returns ERR_OK)

Definition at line 305 of file ip.c.

5.50.2.2 `err_t ip_output (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t tll, u8_t tos, u8_t proto)`

Simple interface to `ip_output_if`. It finds the outgoing network interface and calls upon `ip_output_if` to do the actual work.

Parameters

<i>p</i>	the packet to send (p->payload points to the data, e.g. next protocol header; if dest == IP_HDRINCL, p already includes an IP header and p->payload points to that IP header)
<i>src</i>	the source IP address to send from (if src == IP_ADDR_ANY, the IP address of the netif used to send is used as source address)
<i>dest</i>	the destination IP address to send the packet to
<i>ttl</i>	the TTL value to be set in the IP header
<i>tos</i>	the TOS value to be set in the IP header
<i>proto</i>	the PROTOCOL to be set in the IP header

Returns

ERR_RTE if no route is found see [ip_output_if\(\)](#) for more return values

Definition at line 818 of file ip.c.

5.50.2.3 `err_t ip_output_if (struct pbuf * p, ip_addr_t * src, ip_addr_t * dest, u8_t ttl, u8_t tos, u8_t proto, struct netif * netif)`

Sends an IP packet on a network interface. This function constructs the IP header and calculates the IP header checksum. If the source IP address is NULL, the IP address of the outgoing network interface is filled in as source address. If the destination IP address is IP_HDRINCL, p is assumed to already include an IP header and p->payload points to it instead of the data.

Parameters

<i>p</i>	the packet to send (p->payload points to the data, e.g. next protocol header; if dest == IP_HDRINCL, p already includes an IP header and p->payload points to that IP header)
<i>src</i>	the source IP address to send from (if src == IP_ADDR_ANY, the IP address of the netif used to send is used as source address)
<i>dest</i>	the destination IP address to send the packet to
<i>ttl</i>	the TTL value to be set in the IP header
<i>tos</i>	the TOS value to be set in the IP header
<i>proto</i>	the PROTOCOL to be set in the IP header
<i>netif</i>	the netif on which to send this packet

Returns

ERR_OK if the packet was sent OK ERR_BUF if p doesn't have enough space for IP/LINK headers returns errors returned by netif->output

Note

ip_id: RFC791 "some host may be able to simply use unique identifiers independent of destination"

Definition at line 641 of file ip.c.

5.50.2.4 `struct netif* ip_route (ip_addr_t * dest)`

Finds the appropriate network interface for a given IP address. It searches the list of network interfaces linearly. A match is found if the masked IP address of the network interface equals the masked IP address given to the function.

Parameters

<i>dest</i>	the destination IP address for which to find the route
-------------	--

Returns

the netif on which to send to reach dest

Definition at line 124 of file ip.c.

5.50.3 Variable Documentation**5.50.3.1 const struct ip_hdr* current_header**

Header of the input packet currently being processed.

Definition at line 105 of file ip.c.

5.50.3.2 ip_addr_t current_iphdr_dest

Destination IP address of current_header

Definition at line 109 of file ip.c.

5.50.3.3 ip_addr_t current_iphdr_src

Source IP address of current_header

Definition at line 107 of file ip.c.

5.50.3.4 struct netif* current_netif

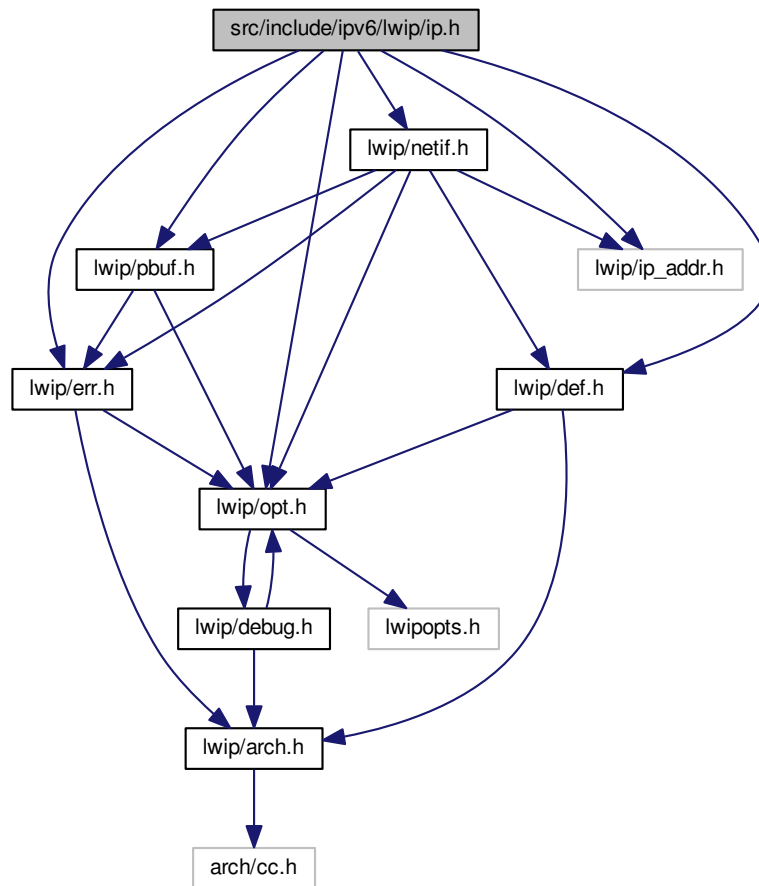
The interface that provided the packet for the current callback invocation.

Definition at line 100 of file ip.c.

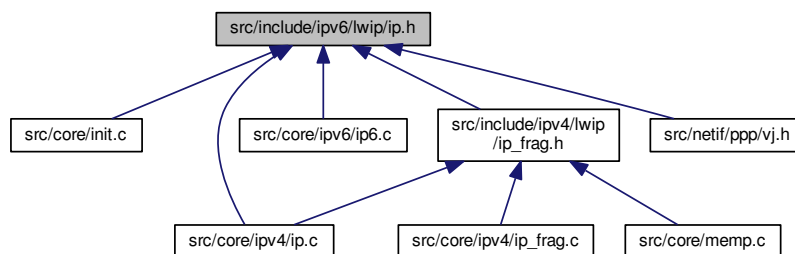
5.50.3.5 PACK_STRUCT_BEGIN struct ip_hdr PACK_STRUCT_STRUCT**5.51 src/include/ipv6/lwip/ip.h File Reference**

```
#include "lwip/opt.h"
#include "lwip/def.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
#include "lwip/err.h"
#include "lwip/netif.h"
```

Include dependency graph for ip.h:



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



Data Structures

- struct [ip_hdr](#)

Macros

- `#define IP_HLEN 40`
- `#define IP_PROTO_ICMP 58`
- `#define IP_PROTO_UDP 17`
- `#define IP_PROTO_UDPLITE 136`
- `#define IP_PROTO_TCP 6`
- `#define IP_HDRINCL NULL`
- `#define IP_PCB_ADDRHINT`
- `#define IP_PCB`
- `#define IPH_PROTO(hdr) (iphdr->nexthdr)`
- `#define ip_current_netif() NULL`
- `#define ip_current_header() NULL`

Functions

- `void ip_init (void)`
- `struct netif * ip_route (struct ip_addr *dest)`
- `void ip_input (struct pbuf *p, struct netif *inp)`
- `err_t ip_output (struct pbuf *p, struct ip_addr *src, struct ip_addr *dest, u8_t ttl, u8_t proto)`
- `err_t ip_output_if (struct pbuf *p, struct ip_addr *src, struct ip_addr *dest, u8_t ttl, u8_t proto, struct netif *netif)`

5.51.1 Macro Definition Documentation

5.51.1.1 `#define ip_current_header() NULL`

Definition at line 118 of file ip.h.

5.51.1.2 `#define ip_current_netif() NULL`

Definition at line 117 of file ip.h.

5.51.1.3 `#define IP_HDRINCL NULL`

Definition at line 59 of file ip.h.

5.51.1.4 `#define IP_HLEN 40`

Definition at line 46 of file ip.h.

5.51.1.5 `#define IP_PCB`

Value:

```
struct ip_addr local_ip; \
  struct ip_addr remote_ip; \
  /* Socket options */ \
  u16_t so_options; \
  /* Type Of Service */ \
  u8_t tos; \
  /* Time To Live */ \
  u8_t ttl; \
  /* link layer address resolution hint */ \
  IP_PCB_ADDRHINT
```

Definition at line 71 of file ip.h.

5.51.1.6 #define IP_PCB_ADDRHINT

Definition at line 64 of file ip.h.

5.51.1.7 #define IP_PROTO_ICMP 58

Definition at line 48 of file ip.h.

5.51.1.8 #define IP_PROTO_TCP 6

Definition at line 51 of file ip.h.

5.51.1.9 #define IP_PROTO_UDP 17

Definition at line 49 of file ip.h.

5.51.1.10 #define IP_PROTO_UDPLITE 136

Definition at line 50 of file ip.h.

5.51.1.11 #define IPH_PROTO(*hdr*) (iphdr->nexthdr)

Definition at line 99 of file ip.h.

5.51.2 Function Documentation**5.51.2.1 void ip_init (void)**

Definition at line 63 of file ip6.c.

5.51.2.2 void ip_input (struct pbuf * *p*, struct netif * *inp*)

This function is called by the network interface device driver when an IP packet is received. The function does the basic checks of the IP header such as packet size being at least larger than the header size etc. If the packet was not destined for us, the packet is forwarded (using ip_forward). The IP checksum is always checked.

Finally, the packet is sent to the upper layer protocol input function.

Parameters

<i>p</i>	the received IP packet (<i>p</i> ->payload points to IP header)
<i>inp</i>	the netif on which this packet was received

Returns

ERR_OK if the packet was processed (could return ERR_* if it wasn't processed, but currently always returns ERR_OK)

Definition at line 305 of file ip.c.

5.51.2.3 err_t ip_output (struct pbuf * *p*, struct ip_addr * *src*, struct ip_addr * *dest*, u8_t *ttl*, u8_t *proto*)

Definition at line 317 of file ip6.c.

5.51.2.4 `err_t ip_output_if (struct pbuf * p, struct ip_addr * src, struct ip_addr * dest, u8_t ttl, u8_t proto, struct netif * netif)`

Definition at line 260 of file ip6.c.

5.51.2.5 `struct netif* ip_route (struct ip_addr * dest)`

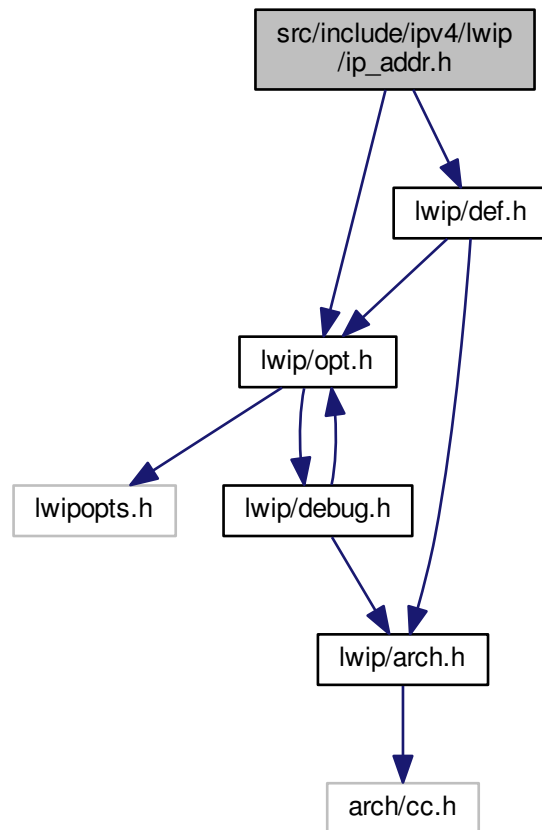
Definition at line 75 of file ip6.c.

5.52 src/include/ipv4/lwip/ip_addr.h File Reference

```
#include "lwip/opt.h"
```

```
#include "lwip/def.h"
```

Include dependency graph for ip_addr.h:



Data Structures

- struct [ip_addr](#)
- struct [ip_addr_packed](#)
- struct [ip_addr2](#)

Macros

- #define `IP_ADDR_ANY` `((ip_addr_t *)&ip_addr_any)`
- #define `IP_ADDR_BROADCAST` `((ip_addr_t *)&ip_addr_broadcast)`
- #define `IPADDR_NONE` `((u32_t)0xffffffffUL)`
- #define `IPADDR_LOOPBACK` `((u32_t)0x7f000001UL)`
- #define `IPADDR_ANY` `((u32_t)0x00000000UL)`
- #define `IPADDR_BROADCAST` `((u32_t)0xffffffffUL)`
- #define `IP_CLASSA(a)` `(((((u32_t)(a)) & 0x80000000UL) == 0)`
- #define `IP_CLASSA_NET` `0xff000000`
- #define `IP_CLASSA_NSHIFT` `24`
- #define `IP_CLASSA_HOST` `(0xffffffff & ~IP_CLASSA_NET)`
- #define `IP_CLASSA_MAX` `128`
- #define `IP_CLASSB(a)` `(((((u32_t)(a)) & 0xc0000000UL) == 0x80000000UL)`
- #define `IP_CLASSB_NET` `0xffff0000`
- #define `IP_CLASSB_NSHIFT` `16`
- #define `IP_CLASSB_HOST` `(0xfffffff & ~IP_CLASSB_NET)`
- #define `IP_CLASSB_MAX` `65536`
- #define `IP_CLASSC(a)` `(((((u32_t)(a)) & 0xe0000000UL) == 0xc0000000UL)`
- #define `IP_CLASSC_NET` `0xfffff000`
- #define `IP_CLASSC_NSHIFT` `8`
- #define `IP_CLASSC_HOST` `(0xfffffff & ~IP_CLASSC_NET)`
- #define `IP_CLASSD(a)` `(((((u32_t)(a)) & 0xf0000000UL) == 0xe0000000UL)`
- #define `IP_CLASSD_NET` `0xf0000000 /* These ones aren't really */`
- #define `IP_CLASSD_NSHIFT` `28 /* net and host fields, but */`
- #define `IP_CLASSD_HOST` `0x0fffffff /* routing needn't know. */`
- #define `IP_MULTICAST(a)` `IP_CLASSD(a)`
- #define `IP_EXPERIMENTAL(a)` `(((((u32_t)(a)) & 0xf0000000UL) == 0xf0000000UL)`
- #define `IP_BADCLASS(a)` `(((((u32_t)(a)) & 0xf0000000UL) == 0xf0000000UL)`
- #define `IP_LOOPBACKNET` `127 /* official! */`
- #define `IP4_ADDR(ipaddr, a, b, c, d)`
- #define `IPADDR2_COPY(dest, src)` `SMEMCPY(dest, src, sizeof(ip_addr_t))`
- #define `ip_addr_copy(dest, src)` `((dest).addr = (src).addr)`
- #define `ip_addr_set(dest, src)`
- #define `ip_addr_set_zero(ipaddr)` `((ipaddr)->addr = 0)`
- #define `ip_addr_set_any(ipaddr)` `((ipaddr)->addr = IPADDR_ANY)`
- #define `ip_addr_set_loopback(ipaddr)` `((ipaddr)->addr = PP_HTONL(IPADDR_LOOPBACK))`
- #define `ip_addr_set_hton(dest, src)`
- #define `ip4_addr_set_u32(dest_ipaddr, src_u32)` `((dest_ipaddr)->addr = (src_u32))`
- #define `ip4_addr_get_u32(src_ipaddr)` `((src_ipaddr)->addr)`
- #define `ip_addr_get_network(target, host, netmask)` `((target)->addr = ((host)->addr) & ((netmask)->addr))`
- #define `ip_addr_netcmp(addr1, addr2, mask)`
- #define `ip_addr_cmp(addr1, addr2)` `((addr1)->addr == (addr2)->addr)`
- #define `ip_addr_isany(addr1)` `((addr1) == NULL || (addr1)->addr == IPADDR_ANY)`
- #define `ip_addr_isbroadcast(ipaddr, netif)` `ip4_addr_isbroadcast((ipaddr)->addr, (netif))`
- #define `ip_addr_netmask_valid(netmask)` `ip4_addr_netmask_valid((netmask)->addr)`
- #define `ip_addr_ismulticast(addr1)` `((((addr1)->addr & PP_HTONL(0xf0000000UL)) == PP_HTONL(0xe0000000UL))`
- #define `ip_addr_islinklocal(addr1)` `((((addr1)->addr & PP_HTONL(0xffff0000UL)) == PP_HTONL(0xa9fe0000UL))`
- #define `ip_addr_debug_print(debug, ipaddr)`
- #define `ip4_addr1(ipaddr)` `((((u8_t*)(ipaddr))[0])`
- #define `ip4_addr2(ipaddr)` `((((u8_t*)(ipaddr))[1])`
- #define `ip4_addr3(ipaddr)` `((((u8_t*)(ipaddr))[2])`
- #define `ip4_addr4(ipaddr)` `((((u8_t*)(ipaddr))[3])`

- `#define ip4_addr1_16(ipaddr) ((u16_t)ip4_addr1(ipaddr))`
- `#define ip4_addr2_16(ipaddr) ((u16_t)ip4_addr2(ipaddr))`
- `#define ip4_addr3_16(ipaddr) ((u16_t)ip4_addr3(ipaddr))`
- `#define ip4_addr4_16(ipaddr) ((u16_t)ip4_addr4(ipaddr))`
- `#define ip_ntoa(ipaddr) ipaddr_ntoa(ipaddr)`

Typedefs

- `typedef typedefPACK_STRUCT_END struct ip_addr ip_addr_t`
- `typedef struct ip_addr_packed ip_addr_p_t`

Functions

- `u8_t ip4_addr_isbroadcast (u32_t addr, const struct netif *netif)`
- `u8_t ip4_addr_netmask_valid (u32_t netmask)`
- `u32_t ipaddr_addr (const char *cp)`
- `int ipaddr_aton (const char *cp, ip_addr_t *addr)`
- `char * ipaddr_ntoa (const ip_addr_t *addr)`
- `char * ipaddr_ntoa_r (const ip_addr_t *addr, char *buf, int buflen)`

Variables

- `PACK_STRUCT_BEGIN struct ip_addr_packed PACK_STRUCT_STRUCT`
- `const ip_addr_t ip_addr_any`
- `const ip_addr_t ip_addr_broadcast`

5.52.1 Macro Definition Documentation

5.52.1.1 `#define IP4_ADDR(ipaddr, a, b, c, d)`

Value:

```
(ipaddr)->addr = ((u32_t) ((a) & 0xff) << 24) | \
                ((u32_t) ((b) & 0xff) << 16) | \
                ((u32_t) ((c) & 0xff) << 8)  | \
                (u32_t) ((d) & 0xff)
```

Set an IP address given by the four byte-parts

Definition at line 139 of file ip_addr.h.

5.52.1.2 `#define ip4_addr1(ipaddr) (((u8_t*)(ipaddr))[0])`

Definition at line 220 of file ip_addr.h.

5.52.1.3 `#define ip4_addr1_16(ipaddr) ((u16_t)ip4_addr1(ipaddr))`

Definition at line 226 of file ip_addr.h.

5.52.1.4 `#define ip4_addr2(ipaddr) (((u8_t*)(ipaddr))[1])`

Definition at line 221 of file ip_addr.h.

5.52.1.5 `#define ip4_addr2_16(ipaddr) ((u16_t)ip4_addr2(ipaddr))`

Definition at line 227 of file ip_addr.h.

5.52.1.6 `#define ip4_addr3(ipaddr) (((u8_t*)(ipaddr))[2])`

Definition at line 222 of file ip_addr.h.

5.52.1.7 `#define ip4_addr3_16(ipaddr) ((u16_t)ip4_addr3(ipaddr))`

Definition at line 228 of file ip_addr.h.

5.52.1.8 `#define ip4_addr4(ipaddr) (((u8_t*)(ipaddr))[3])`

Definition at line 223 of file ip_addr.h.

5.52.1.9 `#define ip4_addr4_16(ipaddr) ((u16_t)ip4_addr4(ipaddr))`

Definition at line 229 of file ip_addr.h.

5.52.1.10 `#define ip4_addr_get_u32(src_ipaddr) ((src_ipaddr)->addr)`

IPv4 only: get the IP address as an u32_t

Definition at line 181 of file ip_addr.h.

5.52.1.11 `#define ip4_addr_set_u32(dest_ipaddr, src_u32) ((dest_ipaddr)->addr = (src_u32))`

IPv4 only: set the IP address given as an u32_t

Definition at line 179 of file ip_addr.h.

5.52.1.12 `#define IP_ADDR_ANY ((ip_addr_t *)&ip_addr_any)`

IP_ADDR_ can be used as a fixed IP address for the wildcard and the broadcast address

Definition at line 92 of file ip_addr.h.

5.52.1.13 `#define IP_ADDR_BROADCAST ((ip_addr_t *)&ip_addr_broadcast)`

Definition at line 93 of file ip_addr.h.

5.52.1.14 `#define ip_addr_cmp(addr1, addr2) ((addr1)->addr == (addr2)->addr)`

Definition at line 198 of file ip_addr.h.

5.52.1.15 `#define ip_addr_copy(dest, src) ((dest).addr = (src).addr)`

Copy IP address - faster than ip_addr_set: no NULL check

Definition at line 162 of file ip_addr.h.

5.52.1.16 #define ip_addr_debug_print(*debug*, *ipaddr*)

Value:

```
LWIP_DEBUGF(debug, ("%U16_F".%"U16_F".%"U16_F".%"U16_F",
                    ipaddr != NULL ? ip4_addr1_16(ipaddr) : 0,
                    ipaddr != NULL ? ip4_addr2_16(ipaddr) : 0,
                    ipaddr != NULL ? ip4_addr3_16(ipaddr) : 0,
                    ipaddr != NULL ? ip4_addr4_16(ipaddr) : 0))
```

Definition at line 212 of file ip_addr.h.

5.52.1.17 #define ip_addr_get_network(*target*, *host*, *netmask*) ((target)->addr = ((host)->addr) & ((netmask)->addr))

Get the network address by combining host address with netmask

Definition at line 184 of file ip_addr.h.

5.52.1.18 #define ip_addr_isany(*addr1*) ((addr1) == NULL || (addr1)->addr == IPADDR_ANY)

Definition at line 200 of file ip_addr.h.

5.52.1.19 #define ip_addr_isbroadcast(*ipaddr*, *netif*) ip4_addr_isbroadcast((ipaddr)->addr, (netif))

Definition at line 202 of file ip_addr.h.

5.52.1.20 #define ip_addr_islinklocal(*addr1*) (((addr1)->addr & PP_HTONL(0xffff0000UL)) == PP_HTONL(0xa9fe0000UL))

Definition at line 210 of file ip_addr.h.

5.52.1.21 #define ip_addr_ismulticast(*addr1*) (((addr1)->addr & PP_HTONL(0xf0000000UL)) == PP_HTONL(0xe0000000UL))

Definition at line 208 of file ip_addr.h.

5.52.1.22 #define ip_addr_netcmp(*addr1*, *addr2*, *mask*)

Value:

```
(( (addr1)->addr & \
                                     (mask)->addr) == \
  ( (addr2)->addr & \
                                     (mask)->addr) )
```

Determine if two address are on the same network.

- *addr1* IP address 1
- *addr2* IP address 2
- *mask* network identifier mask

Returns

!0 if the network identifiers of both address match

Definition at line 194 of file ip_addr.h.

5.52.1.23 `#define ip_addr_netmask_valid(netmask) ip4_addr_netmask_valid((netmask)->addr)`

Definition at line 205 of file ip_addr.h.

5.52.1.24 `#define ip_addr_set(dest, src)`

Value:

```
((dest)->addr = \
                                ((src) == NULL ? 0 : \
                                (src)->addr))
```

Safely copy one IP address to another (src may be NULL)

Definition at line 164 of file ip_addr.h.

5.52.1.25 `#define ip_addr_set_any(ipaddr) ((ipaddr)->addr = IPADDR_ANY)`

Set address to IPADDR_ANY (no need for [htonl\(\)](#))

Definition at line 170 of file ip_addr.h.

5.52.1.26 `#define ip_addr_set_hton(dest, src)`

Value:

```
((dest)->addr = \
                                ((src) == NULL ? 0 : \
                                htonl((src)->
                                addr)))
```

Safely copy one IP address to another and change byte order from host- to network-order.

Definition at line 175 of file ip_addr.h.

5.52.1.27 `#define ip_addr_set_loopback(ipaddr) ((ipaddr)->addr = PP_HTONL(IPADDR_LOOPBACK))`

Set address to loopback address

Definition at line 172 of file ip_addr.h.

5.52.1.28 `#define ip_addr_set_zero(ipaddr) ((ipaddr)->addr = 0)`

Set complete address to zero

Definition at line 168 of file ip_addr.h.

5.52.1.29 `#define IP_BADCLASS(a) (((u32_t)(a) & 0xf0000000UL) == 0xf0000000UL)`

Definition at line 132 of file ip_addr.h.

5.52.1.30 `#define IP_CLASSA(a) (((u32_t)(a)) & 0x80000000UL) == 0)`

Definition at line 108 of file ip_addr.h.

5.52.1.31 `#define IP_CLASSA_HOST (0xffffffff & ~IP_CLASSA_NET)`

Definition at line 111 of file `ip_addr.h`.

5.52.1.32 `#define IP_CLASSA_MAX 128`

Definition at line 112 of file `ip_addr.h`.

5.52.1.33 `#define IP_CLASSA_NET 0xff000000`

Definition at line 109 of file `ip_addr.h`.

5.52.1.34 `#define IP_CLASSA_NSIFT 24`

Definition at line 110 of file `ip_addr.h`.

5.52.1.35 `#define IP_CLASSB(a) (((u32_t)(a)) & 0xc0000000UL) == 0x80000000UL)`

Definition at line 114 of file `ip_addr.h`.

5.52.1.36 `#define IP_CLASSB_HOST (0xffffffff & ~IP_CLASSB_NET)`

Definition at line 117 of file `ip_addr.h`.

5.52.1.37 `#define IP_CLASSB_MAX 65536`

Definition at line 118 of file `ip_addr.h`.

5.52.1.38 `#define IP_CLASSB_NET 0xffff0000`

Definition at line 115 of file `ip_addr.h`.

5.52.1.39 `#define IP_CLASSB_NSIFT 16`

Definition at line 116 of file `ip_addr.h`.

5.52.1.40 `#define IP_CLASSC(a) (((u32_t)(a)) & 0xe0000000UL) == 0xc0000000UL)`

Definition at line 120 of file `ip_addr.h`.

5.52.1.41 `#define IP_CLASSC_HOST (0xffffffff & ~IP_CLASSC_NET)`

Definition at line 123 of file `ip_addr.h`.

5.52.1.42 `#define IP_CLASSC_NET 0xfffff000`

Definition at line 121 of file `ip_addr.h`.

5.52.1.43 `#define IP_CLASSC_NSHIFT 8`

Definition at line 122 of file ip_addr.h.

5.52.1.44 `#define IP_CLASSD(a) (((u32_t)(a) & 0xf0000000UL) == 0xe0000000UL)`

Definition at line 125 of file ip_addr.h.

5.52.1.45 `#define IP_CLASSD_HOST 0x0ffffff /* routing needn't know. */`

Definition at line 128 of file ip_addr.h.

5.52.1.46 `#define IP_CLASSD_NET 0xf0000000 /* These ones aren't really */`

Definition at line 126 of file ip_addr.h.

5.52.1.47 `#define IP_CLASSD_NSHIFT 28 /* net and host fields, but */`

Definition at line 127 of file ip_addr.h.

5.52.1.48 `#define IP_EXPERIMENTAL(a) (((u32_t)(a) & 0xf0000000UL) == 0xf0000000UL)`

Definition at line 131 of file ip_addr.h.

5.52.1.49 `#define IP_LOOPBACKNET 127 /* official! */`

Definition at line 134 of file ip_addr.h.

5.52.1.50 `#define IP_MULTICAST(a) IP_CLASSD(a)`

Definition at line 129 of file ip_addr.h.

5.52.1.51 `#define ip_ntoa(ipaddr) ipaddr_ntoa(ipaddr)`

For backwards compatibility

Definition at line 232 of file ip_addr.h.

5.52.1.52 `#define IPADDR2_COPY(dest, src) SMEMCPY(dest,src,sizeof(ip_addr_t))`

MEMCPY-like copying of IP addresses where addresses are known to be 16-bit-aligned if the port is correctly configured (so a port could define this to copying 2 u16_t's) - no NULL-pointer-checking needed.

Definition at line 158 of file ip_addr.h.

5.52.1.53 `#define IPADDR_ANY ((u32_t)0x00000000UL)`

0.0.0.0

Definition at line 100 of file ip_addr.h.

5.52.1.54 `#define IPADDR_BROADCAST ((u32_t)0xffffffffUL)`

255.255.255.255

Definition at line 102 of file `ip_addr.h`.

5.52.1.55 `#define IPADDR_LOOPBACK ((u32_t)0x7f000001UL)`

127.0.0.1

Definition at line 98 of file `ip_addr.h`.

5.52.1.56 `#define IPADDR_NONE ((u32_t)0xffffffffUL)`

255.255.255.255

Definition at line 96 of file `ip_addr.h`.

5.52.2 Typedef Documentation

5.52.2.1 `typedef struct ip_addr_packed ip_addr_p_t`

Definition at line 65 of file `ip_addr.h`.

5.52.2.2 `typedef typedefPACK_STRUCT_END struct ip_addr ip_addr_t`

`ip_addr_t` uses a struct for convenience only, so that the same defines can operate both on `ip_addr_t` as well as on `ip_addr_p_t`.

Definition at line 64 of file `ip_addr.h`.

5.52.3 Function Documentation

5.52.3.1 `u8_t ip4_addr_isbroadcast (u32_t addr, const struct netif * netif)`

Determine if an address is a broadcast address on a network interface

Parameters

<i>addr</i>	address to be checked
<i>netif</i>	the network interface against which the address is checked

Returns

returns non-zero if the address is a broadcast address

Definition at line 55 of file `ip_addr.c`.

5.52.3.2 `u8_t ip4_addr_netmask_valid (u32_t netmask)`

Checks if a netmask is valid (starting with ones, then only zeros)

Parameters

<i>netmask</i>	the IPv4 netmask to check (in network byte order!)
----------------	--

Returns

1 if the netmask is valid, 0 if it is not

Definition at line 90 of file ip_addr.c.

5.52.3.3 u32_t ipaddr_addr (const char * cp)

Ascii internet address interpretation routine. The value returned is in network order.

Parameters

<i>cp</i>	IP address in ascii representation (e.g. "127.0.0.1")
-----------	---

Returns

ip address in network order

Definition at line 130 of file ip_addr.c.

5.52.3.4 int ipaddr_aton (const char * cp, ip_addr_t * addr)

Check whether "cp" is a valid ascii representation of an Internet address and convert to a binary address. Returns 1 if the address is valid, 0 if not. This replaces inet_addr, the return value from which cannot distinguish between failure and a local broadcast address.

Parameters

<i>cp</i>	IP address in ascii representation (e.g. "127.0.0.1")
<i>addr</i>	pointer to which to save the ip address in network order

Returns

1 if cp could be converted to addr, 0 on failure

Definition at line 152 of file ip_addr.c.

5.52.3.5 char* ipaddr_ntoa (const ip_addr_t * addr)

returns ptr to static buffer; not reentrant!

Convert numeric IP address into decimal dotted ASCII representation. returns ptr to static buffer; not reentrant!

Parameters

<i>addr</i>	ip address in network order to convert
-------------	--

Returns

pointer to a global static (!) buffer that holds the ASCII representation of addr

Definition at line 261 of file ip_addr.c.

5.52.3.6 char* ipaddr_ntoa_r (const ip_addr_t * addr, char * buf, int buflen)

Same as ipaddr_ntoa, but reentrant since a user-supplied buffer is used.

Parameters

<i>addr</i>	ip address in network order to convert
<i>buf</i>	target buffer where the string is stored
<i>buflen</i>	length of buf

Returns

either pointer to buf which now holds the ASCII representation of addr or NULL if buf was too small

Definition at line 276 of file ip_addr.c.

5.52.4 Variable Documentation

5.52.4.1 `const ip_addr_t ip_addr_any`

Definition at line 44 of file ip_addr.c.

5.52.4.2 `const ip_addr_t ip_addr_broadcast`

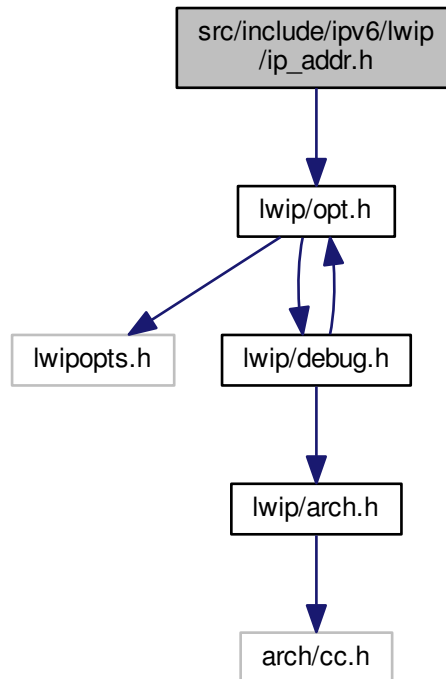
Definition at line 45 of file ip_addr.c.

5.52.4.3 `PACK_STRUCT_BEGIN struct ip_addr2 PACK_STRUCT_STRUCT`

5.53 src/include/ipv6/lwip/ip_addr.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for ip_addr.h:



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



Data Structures

- struct [ip_addr](#)
- struct [ip_addr2](#)

Macros

- `#define` [IP_ADDR_ANY](#) 0
- `#define` [IP6_ADDR](#)(ipaddr, a, b, c, d, e, f, g, h)
- `#define` [ip_addr_debug_print](#)(debug, ipaddr)

Functions

- `u8_t ip_addr_netcmp` (struct `ip_addr` *addr1, struct `ip_addr` *addr2, struct `ip_addr` *mask)
- `u8_t ip_addr_cmp` (struct `ip_addr` *addr1, struct `ip_addr` *addr2)
- `void ip_addr_set` (struct `ip_addr` *dest, struct `ip_addr` *src)
- `u8_t ip_addr_isany` (struct `ip_addr` *addr)

Variables

- `PACK_STRUCT_BEGIN` struct `ip_addr` `PACK_STRUCT_STRUCT`

5.53.1 Macro Definition Documentation

5.53.1.1 `#define IP6_ADDR(ipaddr, a, b, c, d, e, f, g, h)`

Value:

```
do { (ipaddr)->addr[0] = htonl((u32_t)((a & 0xffff) << 16) | (b & 0xffff)); \
      (ipaddr)->addr[1] = htonl((c & 0xffff) << 16) | (d \
      & 0xffff)); \
      (ipaddr)->addr[2] = htonl((e & 0xffff) << 16) | (f \
      & 0xffff)); \
      (ipaddr)->addr[3] = htonl((g & 0xffff) << 16) | (h \
      & 0xffff)); } while(0)
```

Definition at line 71 of file `ip_addr.h`.

5.53.1.2 `#define IP_ADDR_ANY 0`

Definition at line 41 of file `ip_addr.h`.

5.53.1.3 `#define ip_addr_debug_print(debug, ipaddr)`

Value:

```
LWIP_DEBUGF(debug, ("%X32_F":%X32_F":%X32_F":%X32_F":%X32_F":%X32_F":%X32_F":%X32_F"\n", \
      (ntohl(ipaddr->addr[0]) >> 16) & 0xffff, \
      ntohl(ipaddr->addr[0]) & 0xffff, \
      (ntohl(ipaddr->addr[1]) >> 16) & 0xffff, \
      ntohl(ipaddr->addr[1]) & 0xffff, \
      (ntohl(ipaddr->addr[2]) >> 16) & 0xffff, \
      ntohl(ipaddr->addr[2]) & 0xffff, \
      (ntohl(ipaddr->addr[3]) >> 16) & 0xffff, \
      ntohl(ipaddr->addr[3]) & 0xffff));
```

Definition at line 82 of file `ip_addr.h`.

5.53.2 Function Documentation

5.53.2.1 `u8_t ip_addr_cmp(struct ip_addr * addr1, struct ip_addr * addr2)`

Definition at line 49 of file `ip6_addr.c`.

5.53.2.2 `u8_t ip_addr_isany(struct ip_addr * addr)`

Definition at line 68 of file `ip6_addr.c`.

5.53.2.3 `u8_t ip_addr_netcmp (struct ip_addr * addr1, struct ip_addr * addr2, struct ip_addr * mask)`

Definition at line 38 of file ip6_addr.c.

5.53.2.4 `void ip_addr_set (struct ip_addr * dest, struct ip_addr * src)`

Definition at line 58 of file ip6_addr.c.

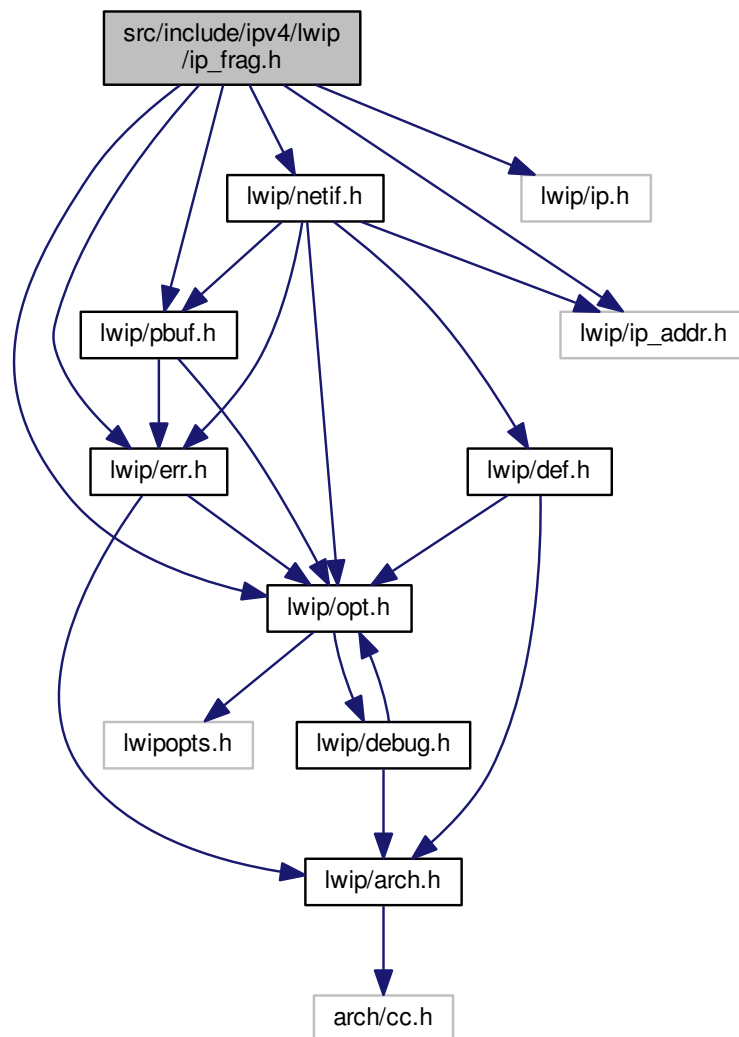
5.53.3 Variable Documentation

5.53.3.1 `PACK_STRUCT_END PACK_STRUCT_BEGIN struct ip_addr2 PACK_STRUCT_STRUCT`

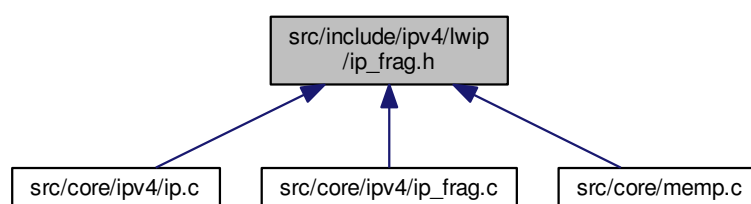
5.54 src/include/ipv4/lwip/ip_frag.h File Reference

```
#include "lwip/opt.h"
#include "lwip/err.h"
#include "lwip/pbuf.h"
#include "lwip/netif.h"
#include "lwip/ip_addr.h"
#include "lwip/ip.h"
```

Include dependency graph for ip_frag.h:



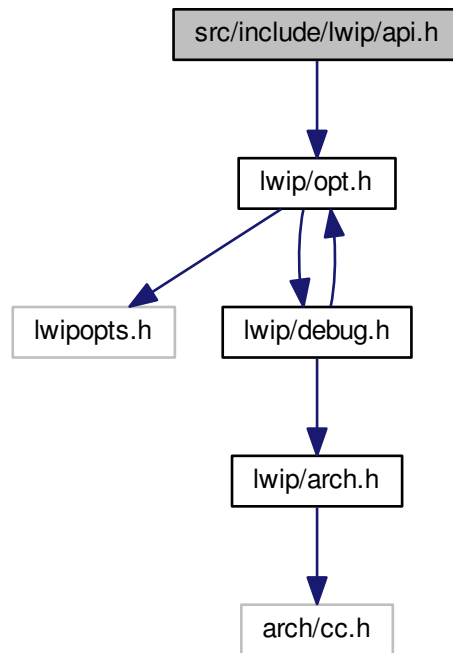
This graph shows which files directly or indirectly include this file:



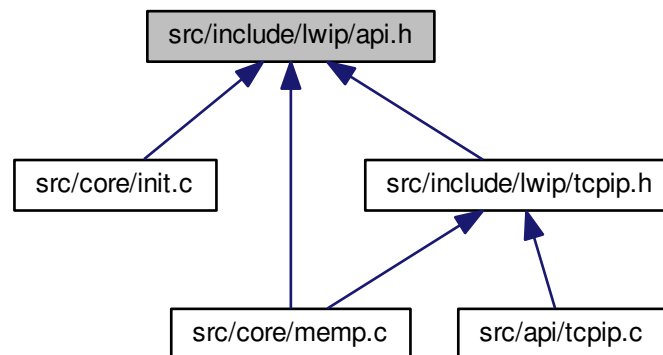
5.55 src/include/lwip/api.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for api.h:



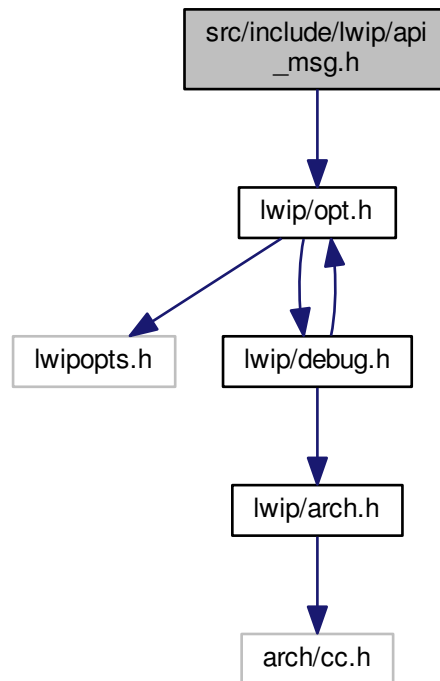
This graph shows which files directly or indirectly include this file:



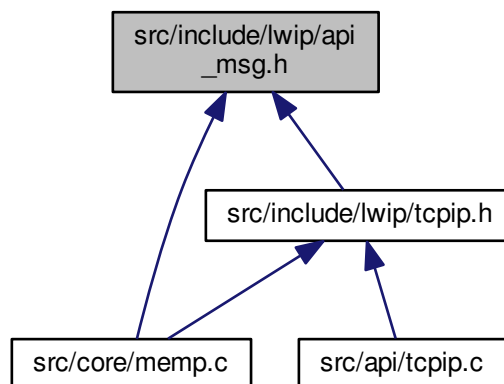
5.56 src/include/lwip/api_msg.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for api_msg.h:



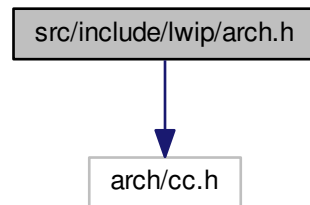
This graph shows which files directly or indirectly include this file:



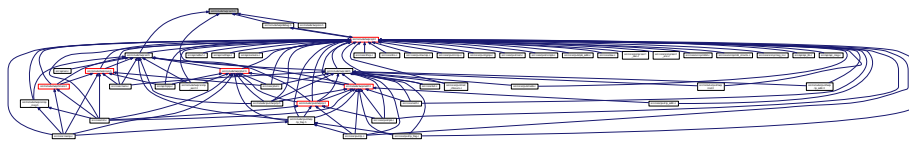
5.57 src/include/lwip/arch.h File Reference

```
#include "arch/cc.h"
```

Include dependency graph for arch.h:



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



Macros

- `#define LITTLE_ENDIAN 1234`
- `#define BIG_ENDIAN 4321`
- `#define SZT_F U32_F`
- `#define X8_F "02x"`
- `#define PACK_STRUCT_BEGIN`
- `#define PACK_STRUCT_END`
- `#define PACK_STRUCT_FIELD(x) x`
- `#define LWIP_UNUSED_ARG(x) (void)x`

5.57.1 Macro Definition Documentation

5.57.1.1 `#define BIG_ENDIAN 4321`

Definition at line 40 of file arch.h.

5.57.1.2 `#define LITTLE_ENDIAN 1234`

Definition at line 36 of file arch.h.

5.57.1.3 `#define LWIP_UNUSED_ARG(x) (void)x`

Definition at line 73 of file arch.h.

5.57.1.4 `#define PACK_STRUCT_BEGIN`

Definition at line 60 of file arch.h.

5.57.1.5 `#define PACK_STRUCT_END`

Definition at line 64 of file arch.h.

5.57.1.6 `#define PACK_STRUCT_FIELD(x) x`

Definition at line 68 of file arch.h.

5.57.1.7 `#define SZT_F U32_F`

Temporary: define format string for size_t if not defined in cc.h

Definition at line 47 of file arch.h.

5.57.1.8 `#define X8_F "02x"`

Temporary upgrade helper: define format string for u8_t as hex if not defined in cc.h

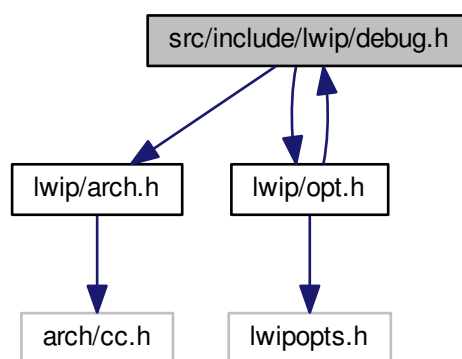
Definition at line 52 of file arch.h.

5.58 `src/include/lwip/debug.h` File Reference

```
#include "lwip/arch.h"
```

```
#include "lwip/opt.h"
```

Include dependency graph for debug.h:



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



Macros

- `#define LWIP_DBG_LEVEL_ALL 0x00`
- `#define LWIP_DBG_LEVEL_OFF LWIP_DBG_LEVEL_ALL /* compatibility define only */`
- `#define LWIP_DBG_LEVEL_WARNING 0x01 /* bad checksums, dropped packets, ... */`
- `#define LWIP_DBG_LEVEL_SERIOUS 0x02 /* memory allocation failures, ... */`
- `#define LWIP_DBG_LEVEL_SEVERE 0x03`
- `#define LWIP_DBG_MASK_LEVEL 0x03`
- `#define LWIP_DBG_ON 0x80U`
- `#define LWIP_DBG_OFF 0x00U`
- `#define LWIP_DBG_TRACE 0x40U`
- `#define LWIP_DBG_STATE 0x20U`
- `#define LWIP_DBG_FRESH 0x10U`
- `#define LWIP_DBG_HALT 0x08U`
- `#define LWIP_ASSERT(message, assertion)`
- `#define LWIP_ERROR(message, expression, handler)`
- `#define LWIP_DEBUGF(debug, message)`

5.58.1 Macro Definition Documentation

5.58.1.1 `#define LWIP_ASSERT(message, assertion)`

Value:

```
do { if(!(assertion)) \
    LWIP_PLATFORM_ASSERT(message); } while(0)
```

Definition at line 66 of file debug.h.

5.58.1.2 `#define LWIP_DBG_FRESH 0x10U`

flag for LWIP_DEBUGF indicating newly added code, not thoroughly tested yet

Definition at line 61 of file debug.h.

5.58.1.3 `#define LWIP_DBG_HALT 0x08U`

flag for LWIP_DEBUGF to halt after printing this debug message

Definition at line 63 of file debug.h.

5.58.1.4 `#define LWIP_DBG_LEVEL_ALL 0x00`

lower two bits indicate debug level

- 0 all
- 1 warning
- 2 serious
- 3 severe

Definition at line 44 of file debug.h.

5.58.1.5 `#define LWIP_DBG_LEVEL_OFF LWIP_DBG_LEVEL_ALL /* compatibility define only */`

Definition at line 45 of file debug.h.

5.58.1.6 `#define LWIP_DBG_LEVEL_SERIOUS 0x02 /* memory allocation failures, ... */`

Definition at line 47 of file debug.h.

5.58.1.7 `#define LWIP_DBG_LEVEL_SEVERE 0x03`

Definition at line 48 of file debug.h.

5.58.1.8 `#define LWIP_DBG_LEVEL_WARNING 0x01 /* bad checksums, dropped packets, ... */`

Definition at line 46 of file debug.h.

5.58.1.9 `#define LWIP_DBG_MASK_LEVEL 0x03`

Definition at line 49 of file debug.h.

5.58.1.10 `#define LWIP_DBG_OFF 0x00U`

flag for LWIP_DEBUGF to disable that debug message

Definition at line 54 of file debug.h.

5.58.1.11 `#define LWIP_DBG_ON 0x80U`

flag for LWIP_DEBUGF to enable that debug message

Definition at line 52 of file debug.h.

5.58.1.12 `#define LWIP_DBG_STATE 0x20U`

flag for LWIP_DEBUGF indicating a state debug message (to follow module states)

Definition at line 59 of file debug.h.

5.58.1.13 `#define LWIP_DBG_TRACE 0x40U`

flag for LWIP_DEBUGF indicating a tracing message (to follow program flow)

Definition at line 57 of file debug.h.

5.58.1.14 `#define LWIP_DEBUGF(debug, message)`

Definition at line 95 of file debug.h.

5.58.1.15 `#define LWIP_ERROR(message, expression, handler)`

Value:

if "expression" isn't true, then print "message" and execute "handler" expression
Definition at line 74 of file debug.h.

```
#include "lwip/arch.h"
#include "lwip/opt.h"
Include dependency graph for def.h:
```



- Generated on Sat Feb 13 2016 16:13:47 for lwIP by Doxygen

- `#define LWIP_MIN(x, y) (((x) < (y)) ? (x) : (y))`
- `#define NULL ((void *)0)`
- `#define LWIP_MAKE_U16(a, b) ((a << 8) | b)`
- `#define LWIP_PLATFORM_BYTESWAP 0`
- `#define htons(x) lwip_htons(x)`
- `#define ntohs(x) lwip_ntohs(x)`
- `#define htonl(x) lwip_htonl(x)`
- `#define ntohl(x) lwip_ntohl(x)`
- `#define lwip_htons(x) (x)`
- `#define lwip_ntohs(x) (x)`
- `#define lwip_htonl(x) (x)`
- `#define lwip_ntohl(x) (x)`
- `#define PP_HTONS(x) (x)`
- `#define PP_NTOHS(x) (x)`
- `#define PP_HTONL(x) (x)`
- `#define PP_NTOHL(x) (x)`

5.59.1 Macro Definition Documentation

5.59.1.1 `#define htonl(x) lwip_htonl(x)`

Definition at line 79 of file def.h.

5.59.1.2 `#define htons(x) lwip_htons(x)`

Definition at line 77 of file def.h.

5.59.1.3 `#define lwip_htonl(x) (x)`

Definition at line 86 of file def.h.

5.59.1.4 `#define lwip_htons(x) (x)`

Definition at line 84 of file def.h.

5.59.1.5 `#define LWIP_MAKE_U16(a, b) ((a << 8) | b)`

Definition at line 52 of file def.h.

5.59.1.6 `#define LWIP_MAX(x, y) (((x) > (y)) ? (x) : (y))`

Definition at line 43 of file def.h.

5.59.1.7 `#define LWIP_MIN(x, y) (((x) < (y)) ? (x) : (y))`

Definition at line 44 of file def.h.

5.59.1.8 `#define lwip_ntohl(x) (x)`

Definition at line 87 of file def.h.

5.59.1.9 `#define lwip_ntohs(x)(x)`

Definition at line 85 of file def.h.

5.59.1.10 `#define LWIP_PLATFORM_BYTESWAP 0`

Definition at line 58 of file def.h.

5.59.1.11 `#define ntohl(x) lwip_ntohl(x)`

Definition at line 80 of file def.h.

5.59.1.12 `#define ntohs(x) lwip_ntohs(x)`

Definition at line 78 of file def.h.

5.59.1.13 `#define NULL ((void *)0)`

Definition at line 47 of file def.h.

5.59.1.14 `#define PP_HTONL(x)(x)`

Definition at line 90 of file def.h.

5.59.1.15 `#define PP_HTONS(x)(x)`

Definition at line 88 of file def.h.

5.59.1.16 `#define PP_NTOHL(x)(x)`

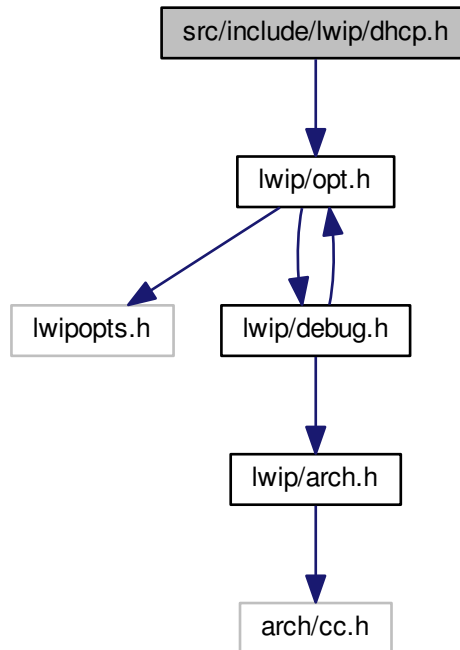
Definition at line 91 of file def.h.

5.59.1.17 `#define PP_NTOHS(x)(x)`

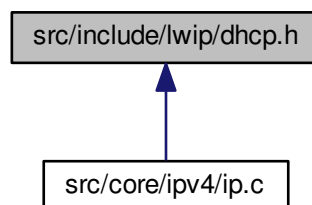
Definition at line 89 of file def.h.

5.60 src/include/lwip/dhcp.h File Reference

```
#include "lwip/opt.h"
Include dependency graph for dhcp.h:
```



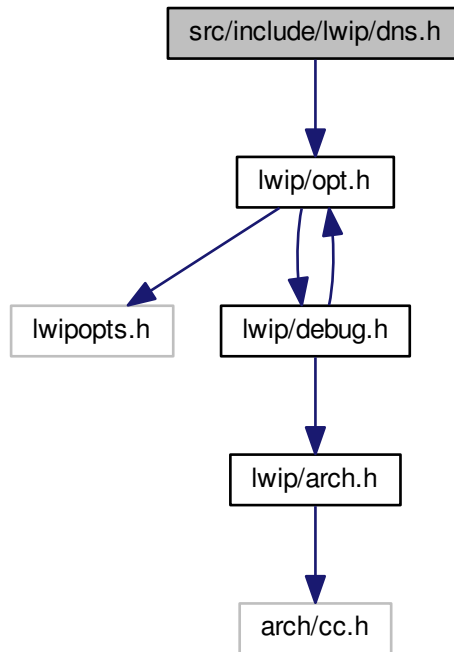
This graph shows which files directly or indirectly include this file:



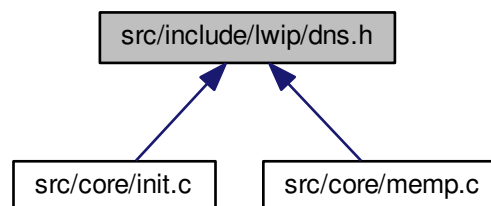
5.61 src/include/lwip/dns.h File Reference

```
#include "lwip/opt.h"
```


Include dependency graph for dns.h:



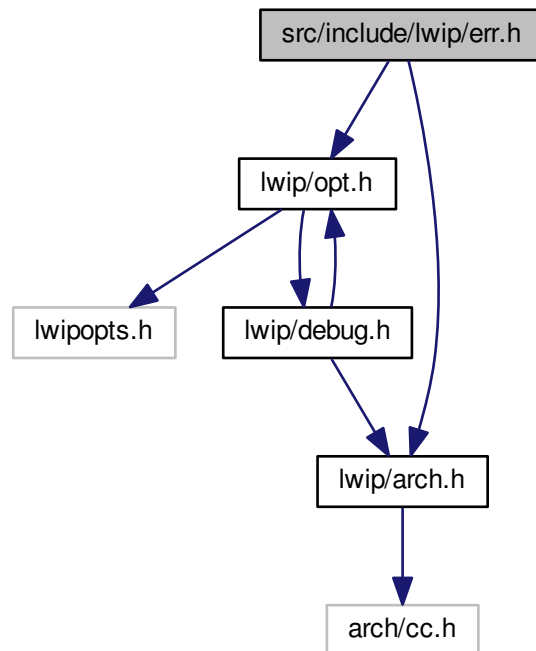
This graph shows which files directly or indirectly include this file:



5.62 src/include/lwip/err.h File Reference

```
#include "lwip/opt.h"
#include "lwip/arch.h"
```

Include dependency graph for err.h:



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



Macros

- `#define ERR_OK 0` /* No error, everything OK. */
- `#define ERR_MEM -1` /* Out of memory error. */
- `#define ERR_BUF -2` /* Buffer error. */
- `#define ERR_TIMEOUT -3` /* Timeout. */
- `#define ERR_RTE -4` /* Routing problem. */
- `#define ERR_INPROGRESS -5` /* Operation in progress */
- `#define ERR_VAL -6` /* Illegal value. */
- `#define ERR_WOULDBLOCK -7` /* Operation would block. */
- `#define ERR_USE -8` /* Address in use. */
- `#define ERR_ISCONN -9` /* Already connected. */
- `#define ERR_IS_FATAL(e) ((e) < ERR_ISCONN)`
- `#define ERR_ABRT -10` /* Connection aborted. */
- `#define ERR_RST -11` /* Connection reset. */
- `#define ERR_CLSD -12` /* Connection closed. */

- `#define ERR_CONN -13 /* Not connected. */`
- `#define ERR_ARG -14 /* Illegal argument. */`
- `#define ERR_IF -15 /* Low-level netif error */`
- `#define lwip_strerror(x) ""`

Typedefs

- `typedef s8_t err_t`

5.62.1 Macro Definition Documentation

5.62.1.1 `#define ERR_ABRT -10 /* Connection aborted. */`

Definition at line 65 of file err.h.

5.62.1.2 `#define ERR_ARG -14 /* Illegal argument. */`

Definition at line 70 of file err.h.

5.62.1.3 `#define ERR_BUF -2 /* Buffer error. */`

Definition at line 54 of file err.h.

5.62.1.4 `#define ERR_CLSD -12 /* Connection closed. */`

Definition at line 67 of file err.h.

5.62.1.5 `#define ERR_CONN -13 /* Not connected. */`

Definition at line 68 of file err.h.

5.62.1.6 `#define ERR_IF -15 /* Low-level netif error */`

Definition at line 72 of file err.h.

5.62.1.7 `#define ERR_INPROGRESS -5 /* Operation in progress */`

Definition at line 57 of file err.h.

5.62.1.8 `#define ERR_IS_FATAL(e) ((e) < ERR_ISCONN)`

Definition at line 63 of file err.h.

5.62.1.9 `#define ERR_ISCONN -9 /* Already connected. */`

Definition at line 61 of file err.h.

5.62.1.10 `#define ERR_MEM -1 /* Out of memory error. */`

Definition at line 53 of file err.h.

5.62.1.11 `#define ERR_OK 0 /* No error, everything OK. */`

Definition at line 52 of file err.h.

5.62.1.12 `#define ERR_RST -11 /* Connection reset. */`

Definition at line 66 of file err.h.

5.62.1.13 `#define ERR_RTE -4 /* Routing problem. */`

Definition at line 56 of file err.h.

5.62.1.14 `#define ERR_TIMEOUT -3 /* Timeout. */`

Definition at line 55 of file err.h.

5.62.1.15 `#define ERR_USE -8 /* Address in use. */`

Definition at line 60 of file err.h.

5.62.1.16 `#define ERR_VAL -6 /* Illegal value. */`

Definition at line 58 of file err.h.

5.62.1.17 `#define ERR_WOULDBLOCK -7 /* Operation would block. */`

Definition at line 59 of file err.h.

5.62.1.18 `#define lwip_strerror(x) ""`

Definition at line 78 of file err.h.

5.62.2 Typedef Documentation

5.62.2.1 `typedef s8_t err_t`

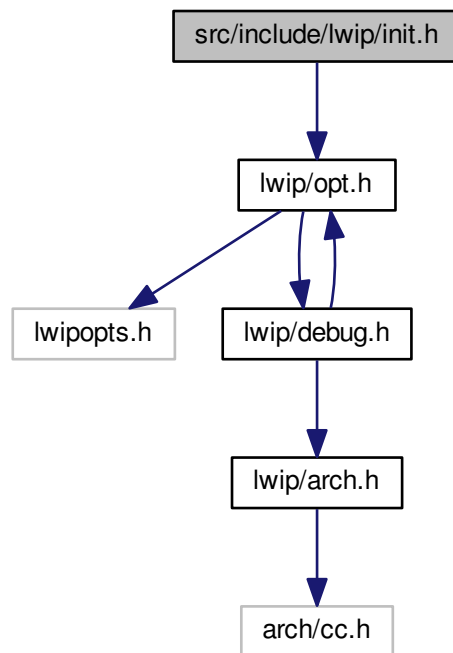
Define LWIP_ERR_T in cc.h if you want to use a different type for your platform (must be signed).

Definition at line 47 of file err.h.

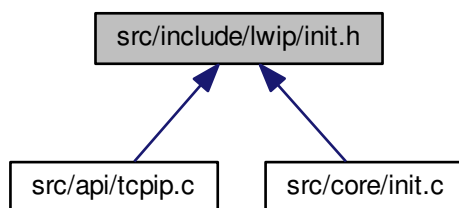
5.63 src/include/lwip/init.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for init.h:



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



Macros

- `#define LWIP_VERSION_MAJOR 1U`
- `#define LWIP_VERSION_MINOR 4U`
- `#define LWIP_VERSION_REVISION 1U`
- `#define LWIP_VERSION_RC 255U`
- `#define LWIP_RC_RELEASE 255U`
- `#define LWIP_RC_DEVELOPMENT 0U`
- `#define LWIP_VERSION_IS_RELEASE (LWIP_VERSION_RC == LWIP_RC_RELEASE)`

- `#define LWIP_VERSION_IS_DEVELOPMENT (LWIP_VERSION_RC == LWIP_RC_DEVELOPMENT)`
- `#define LWIP_VERSION_IS_RC ((LWIP_VERSION_RC != LWIP_RC_RELEASE) && (LWIP_VERSION_↵
RC != LWIP_RC_DEVELOPMENT))`
- `#define LWIP_VERSION`

Functions

- void `lwip_init` (void)

5.63.1 Macro Definition Documentation

5.63.1.1 `#define LWIP_RC_DEVELOPMENT 0U`

LWIP_VERSION_RC is set to LWIP_RC_DEVELOPMENT for CVS versions

Definition at line 55 of file init.h.

5.63.1.2 `#define LWIP_RC_RELEASE 255U`

LWIP_VERSION_RC is set to LWIP_RC_RELEASE for official releases

Definition at line 53 of file init.h.

5.63.1.3 `#define LWIP_VERSION`

Value:

```
(LWIP_VERSION_MAJOR << 24 | LWIP_VERSION_MINOR << 16 |  
    \                               LWIP_VERSION_REVISION << 8 |  
    LWIP_VERSION_RC)
```

Provides the version of the stack

Definition at line 62 of file init.h.

5.63.1.4 `#define LWIP_VERSION_IS_DEVELOPMENT (LWIP_VERSION_RC == LWIP_RC_DEVELOPMENT)`

Definition at line 58 of file init.h.

5.63.1.5 `#define LWIP_VERSION_IS_RC ((LWIP_VERSION_RC != LWIP_RC_RELEASE) && (LWIP_VERSION_RC != LWIP_RC_DEVELOPMENT))`

Definition at line 59 of file init.h.

5.63.1.6 `#define LWIP_VERSION_IS_RELEASE (LWIP_VERSION_RC == LWIP_RC_RELEASE)`

Definition at line 57 of file init.h.

5.63.1.7 `#define LWIP_VERSION_MAJOR 1U`

X.x.x: Major version of the stack

Definition at line 42 of file init.h.

5.63.1.8 `#define LWIP_VERSION_MINOR 4U`

x.X.x: Minor version of the stack

Definition at line 44 of file init.h.

5.63.1.9 `#define LWIP_VERSION_RC 255U`

For release candidates, this is set to 1..254 For official releases, this is set to 255 (LWIP_RC_RELEASE) For development versions (CVS), this is set to 0 (LWIP_RC_DEVELOPMENT)

Definition at line 50 of file init.h.

5.63.1.10 `#define LWIP_VERSION_REVISION 1U`

x.x.X: Revision of the stack

Definition at line 46 of file init.h.

5.63.2 Function Documentation

5.63.2.1 `void lwip_init (void)`

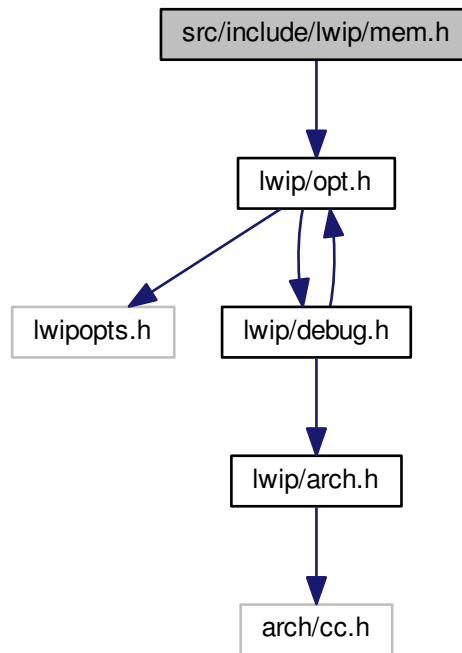
Perform Sanity check of user-configurable values, and initialize all modules.

Definition at line 289 of file init.c.

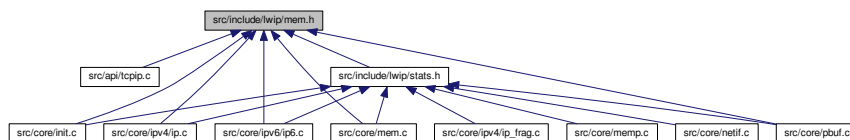
5.64 src/include/lwip/mem.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for mem.h:



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



Macros

- `#define MEM_SIZE_F U16_F`
- `#define LWIP_MEM_ALIGN_SIZE(size) (((size) + MEM_ALIGNMENT - 1) & ~(MEM_ALIGNMENT-1))`
- `#define LWIP_MEM_ALIGN_BUFFER(size) (((size) + MEM_ALIGNMENT - 1))`
- `#define LWIP_MEM_ALIGN(addr) ((void *)(((mem_ptr_t)(addr) + MEM_ALIGNMENT - 1) & ~(mem_ptr_t)(MEM_ALIGNMENT-1)))`

Typedefs

- `typedef u16_t mem_size_t`

Functions

- void [mem_init](#) (void)
- void * [mem_trim](#) (void *mem, [mem_size_t](#) size)
- void * [mem_malloc](#) ([mem_size_t](#) size)
- void * [mem_calloc](#) ([mem_size_t](#) count, [mem_size_t](#) size)
- void [mem_free](#) (void *mem)

5.64.1 Macro Definition Documentation

5.64.1.1 `#define LWIP_MEM_ALIGN(addr) ((void *)(((mem_ptr_t)(addr) + MEM_ALIGNMENT - 1) & ~((mem_ptr_t)(MEM_ALIGNMENT-1)))`

Align a memory pointer to the alignment defined by MEM_ALIGNMENT so that ADDR % MEM_ALIGNMENT == 0
Definition at line 116 of file mem.h.

5.64.1.2 `#define LWIP_MEM_ALIGN_BUFFER(size) (((size) + MEM_ALIGNMENT - 1))`

Calculate safe memory size for an aligned buffer when using an unaligned type as storage. This includes a safety-margin on (MEM_ALIGNMENT - 1) at the start (e.g. if buffer is `u8_t[]` and actual data will be `u32_t*`)

Definition at line 109 of file mem.h.

5.64.1.3 `#define LWIP_MEM_ALIGN_SIZE(size) (((size) + MEM_ALIGNMENT - 1) & ~((MEM_ALIGNMENT-1))`

Calculate memory size for an aligned buffer - returns the next highest multiple of MEM_ALIGNMENT (e.g. [LWIP_MEM_ALIGN_SIZE\(3\)](#) and [LWIP_MEM_ALIGN_SIZE\(4\)](#) will both yield 4 for MEM_ALIGNMENT == 4).

Definition at line 101 of file mem.h.

5.64.1.4 `#define MEM_SIZE_F U16_F`

Definition at line 77 of file mem.h.

5.64.2 Typedef Documentation

5.64.2.1 `typedef u16_t mem_size_t`

Definition at line 76 of file mem.h.

5.64.3 Function Documentation

5.64.3.1 `void* mem_calloc (mem_size_t count, mem_size_t size)`

Contiguously allocates enough space for count objects that are size bytes of memory each and returns a pointer to the allocated memory.

The allocated memory is filled with bytes of value zero.

Parameters

<i>count</i>	number of objects to allocate
<i>size</i>	size of the objects to allocate

Returns

pointer to allocated memory / NULL pointer if there is an error

Definition at line 646 of file mem.c.

5.64.3.2 void mem_free (void * *rmem*)

Put a struct mem back on the heap

Parameters

<i>rmem</i>	is the data portion of a struct mem as returned by a previous call to mem_malloc()
-------------	--

Definition at line 311 of file mem.c.

5.64.3.3 void mem_init (void)

Zero the heap and initialize start, end and lowest-free

Definition at line 274 of file mem.c.

5.64.3.4 void* mem_malloc (mem_size_t *size*)

Adam's [mem_malloc\(\)](#) plus solution for bug #17922 Allocate a block of memory with a minimum of 'size' bytes.

Parameters

<i>size</i>	is the minimum size of the requested block in bytes.
-------------	--

Returns

pointer to allocated memory or NULL if no free memory was found.

Note that the returned value will always be aligned (as defined by MEM_ALIGNMENT).

Definition at line 494 of file mem.c.

5.64.3.5 void* mem_trim (void * *rmem*, mem_size_t *newsize*)

Shrink memory returned by [mem_malloc\(\)](#).

Parameters

<i>rmem</i>	pointer to memory allocated by mem_malloc the is to be shrinked
<i>newsize</i>	required size after shrinking (needs to be smaller than or equal to the previous size)

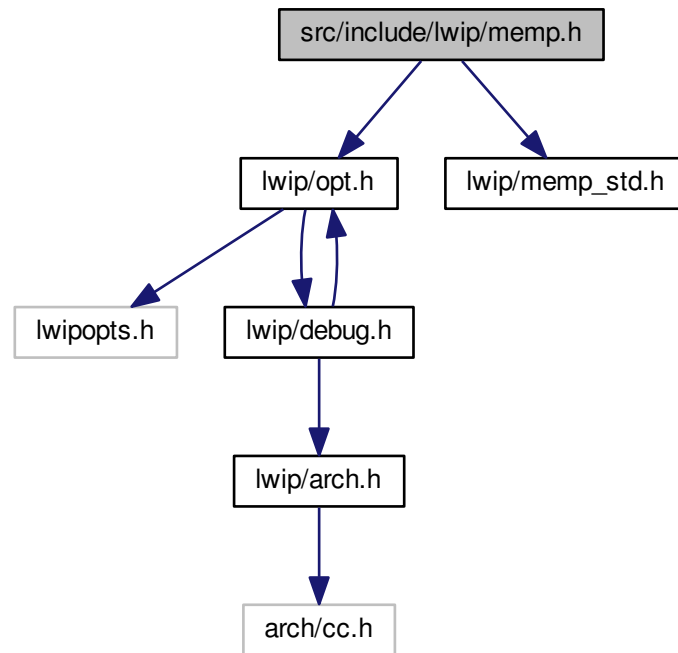
Returns

for compatibility reasons: is always == rmem, at the moment or NULL if newsize is > old size, in which case rmem is NOT touched or freed!

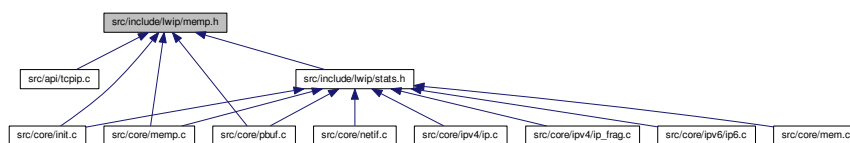
Definition at line 369 of file mem.c.

5.65 src/include/lwip/memp.h File Reference

```
#include "lwip/opt.h"
#include "lwip/memp_std.h"
Include dependency graph for memp.h:
```



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



Macros

- `#define LWIP_MEMPOOL(name, num, size, desc) MEMP_##name,`

Enumerations

- `enum memp_t { MEMP_MAX }`

Functions

- void `memp_init` (void)
- void * `memp_malloc` (`memp_t` type)
- void `memp_free` (`memp_t` type, void *`mem`)

5.65.1 Macro Definition Documentation

5.65.1.1 `#define LWIP_MEMPOOL(name, num, size, desc) MEMP_##name,`

Definition at line 44 of file `memp.h`.

5.65.2 Enumeration Type Documentation

5.65.2.1 `enum memp_t`

Enumerator

MEMP_MAX

Definition at line 43 of file `memp.h`.

5.65.3 Function Documentation

5.65.3.1 `void memp_free (memp_t type, void * mem)`

Put an element back into its pool.

Parameters

<i>type</i>	the pool where to put mem
<i>mem</i>	the memp element to free

Definition at line 435 of file `memp.c`.

5.65.3.2 `void memp_init (void)`

Initialize this module.

Carves out `memp_memory` into linked lists for each pool-type.

Definition at line 338 of file `memp.c`.

5.65.3.3 `void* memp_malloc (memp_t type)`

Get an element from a specific pool.

Parameters

<i>type</i>	the pool to get an element from
-------------	---------------------------------

the debug version has two more parameters:

Parameters

<i>file</i>	file name calling this function
<i>line</i>	number of line where this function is called

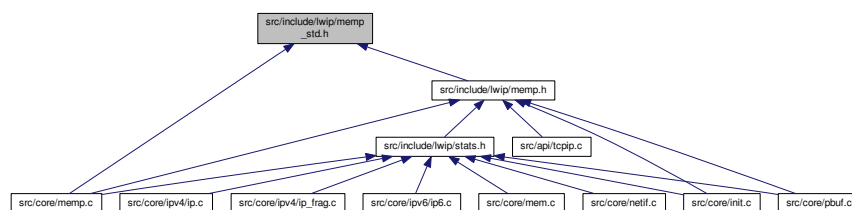
Returns

a pointer to the allocated memory or a NULL pointer on error

Definition at line 390 of file memp.c.

5.66 src/include/lwip/memp_std.h File Reference

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

**Macros**

- `#define LWIP_MALLOC_MEMPOOL(num, size) LWIP_MEMPOOL(POOL_##size, num, (size + sizeof(struct memp_malloc_helper)), "MALLOC_"#size)`
- `#define LWIP_MALLOC_MEMPOOL_START`
- `#define LWIP_MALLOC_MEMPOOL_END`
- `#define LWIP_PBUF_MEMPOOL(name, num, payload, desc) LWIP_MEMPOOL(name, num, (MEMP_ALIGN_SIZE(sizeof(struct pbuf)) + MEMP_ALIGN_SIZE(payload)), desc)`

5.66.1 Macro Definition Documentation

5.66.1.1 `#define LWIP_MALLOC_MEMPOOL(num, size) LWIP_MEMPOOL(POOL_##size, num, (size + sizeof(struct memp_malloc_helper)), "MALLOC_"#size)`

Definition at line 15 of file memp_std.h.

5.66.1.2 `#define LWIP_MALLOC_MEMPOOL_END`

Definition at line 17 of file memp_std.h.

5.66.1.3 `#define LWIP_MALLOC_MEMPOOL_START`

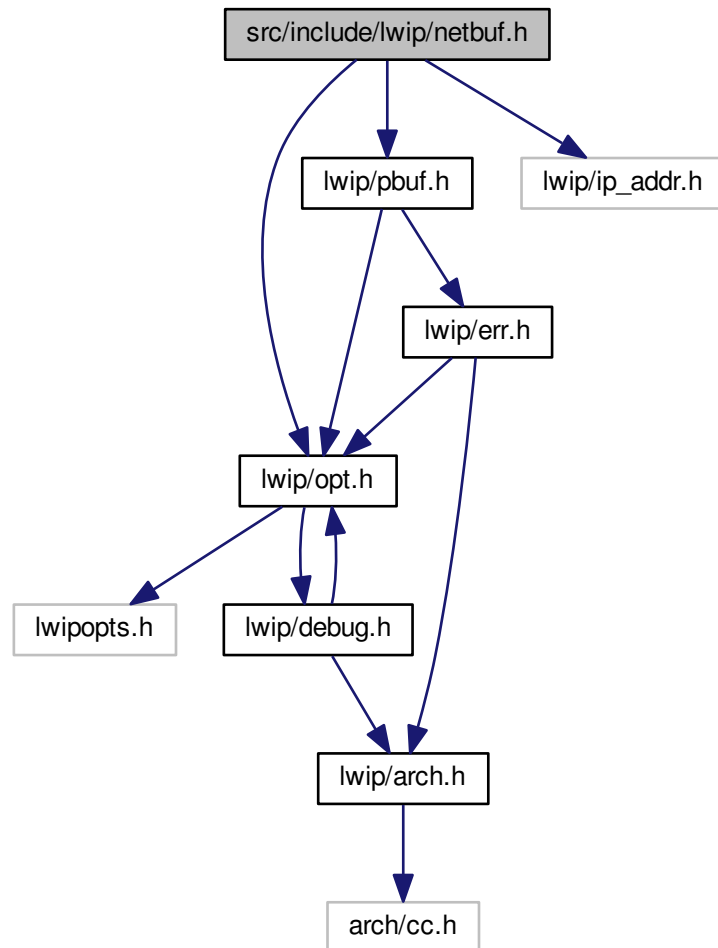
Definition at line 16 of file memp_std.h.

5.66.1.4 `#define LWIP_PBUF_MEMPOOL(name, num, payload, desc) LWIP_MEMPOOL(name, num, (MEMP_ALIGN_SIZE(sizeof(struct pbuf)) + MEMP_ALIGN_SIZE(payload)), desc)`

Definition at line 23 of file memp_std.h.

5.67 src/include/lwip/netbuf.h File Reference

```
#include "lwip/opt.h"
#include "lwip/pbuf.h"
#include "lwip/ip_addr.h"
Include dependency graph for netbuf.h:
```



Data Structures

- struct [netbuf](#)

Macros

- #define [NETBUF_FLAG_DESTADDR](#) 0x01
- #define [NETBUF_FLAG_CHKSUM](#) 0x02
- #define [netbuf_copy_partial](#)(buf, dataptr, len, offset) [pbuf_copy_partial](#)((buf)->p, (dataptr), (len), (offset))
- #define [netbuf_copy](#)(buf, dataptr, len) [netbuf_copy_partial](#)(buf, dataptr, len, 0)
- #define [netbuf_take](#)(buf, dataptr, len) [pbuf_take](#)((buf)->p, dataptr, len)

- `#define netbuf_len(buf) ((buf)->p->tot_len)`
- `#define netbuf_fromaddr(buf) (&((buf)->addr))`
- `#define netbuf_set_fromaddr(buf, fromaddr) ip_addr_set(&((buf)->addr), fromaddr)`
- `#define netbuf_fromport(buf) ((buf)->port)`

Functions

- `struct netbuf * netbuf_new (void)`
- `void netbuf_delete (struct netbuf *buf)`
- `void * netbuf_alloc (struct netbuf *buf, u16_t size)`
- `void netbuf_free (struct netbuf *buf)`
- `err_t netbuf_ref (struct netbuf *buf, const void *dataptr, u16_t size)`
- `void netbuf_chain (struct netbuf *head, struct netbuf *tail)`
- `err_t netbuf_data (struct netbuf *buf, void **dataptr, u16_t *len)`
- `s8_t netbuf_next (struct netbuf *buf)`
- `void netbuf_first (struct netbuf *buf)`

5.67.1 Macro Definition Documentation

5.67.1.1 `#define netbuf_copy(buf, dataptr, len) netbuf_copy_partial(buf, dataptr, len, 0)`

Definition at line 81 of file netbuf.h.

5.67.1.2 `#define netbuf_copy_partial(buf, dataptr, len, offset) pbuf_copy_partial((buf)->p, (dataptr), (len), (offset))`

Definition at line 79 of file netbuf.h.

5.67.1.3 `#define NETBUF_FLAG_CHKSUM 0x02`

This netbuf includes a checksum

Definition at line 46 of file netbuf.h.

5.67.1.4 `#define NETBUF_FLAG_DESTADDR 0x01`

This netbuf has dest-addr/port set

Definition at line 44 of file netbuf.h.

5.67.1.5 `#define netbuf_fromaddr(buf) (&((buf)->addr))`

Definition at line 84 of file netbuf.h.

5.67.1.6 `#define netbuf_fromport(buf) ((buf)->port)`

Definition at line 86 of file netbuf.h.

5.67.1.7 `#define netbuf_len(buf) ((buf)->p->tot_len)`

Definition at line 83 of file netbuf.h.

5.67.1.8 `#define netbuf_set_fromaddr(buf, fromaddr) ip_addr_set(&(buf)->addr, fromaddr)`

Definition at line 85 of file netbuf.h.

5.67.1.9 `#define netbuf_take(buf, dataptr, len) pbuf_take((buf)->p, dataptr, len)`

Definition at line 82 of file netbuf.h.

5.67.2 Function Documentation

5.67.2.1 `void* netbuf_alloc (struct netbuf * buf, u16_t size)`

5.67.2.2 `void netbuf_chain (struct netbuf * head, struct netbuf * tail)`

5.67.2.3 `err_t netbuf_data (struct netbuf * buf, void ** dataptr, u16_t * len)`

5.67.2.4 `void netbuf_delete (struct netbuf * buf)`

5.67.2.5 `void netbuf_first (struct netbuf * buf)`

5.67.2.6 `void netbuf_free (struct netbuf * buf)`

5.67.2.7 `struct netbuf* netbuf_new (void)`

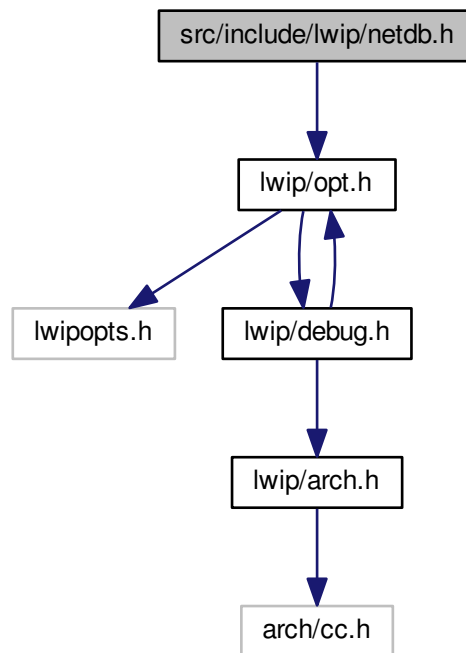
5.67.2.8 `s8_t netbuf_next (struct netbuf * buf)`

5.67.2.9 `err_t netbuf_ref (struct netbuf * buf, const void * dataptr, u16_t size)`

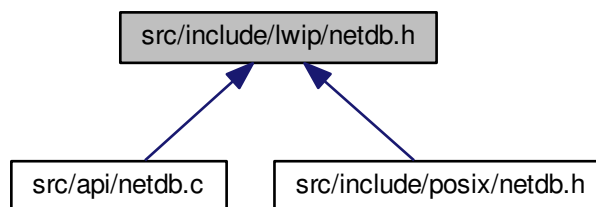
5.68 src/include/lwip/netdb.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for netdb.h:



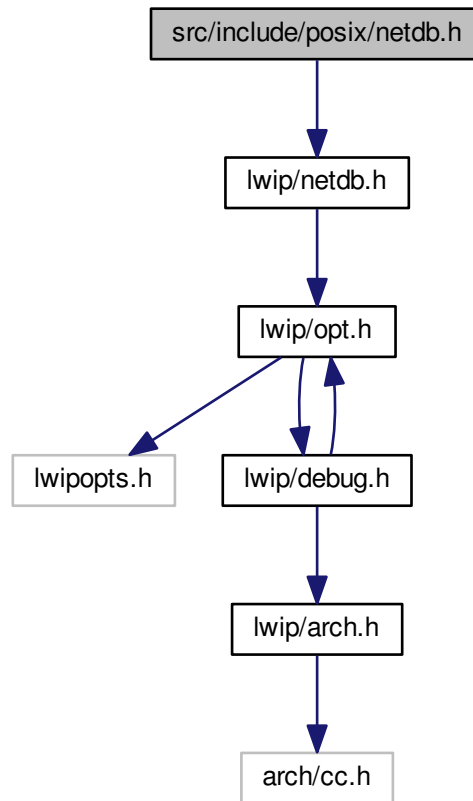
This graph shows which files directly or indirectly include this file:



5.69 src/include/posix/netdb.h File Reference

```
#include "lwip/netdb.h"
```

Include dependency graph for netdb.h:



5.69.1 Detailed Description

This file is a posix wrapper for [lwip/netdb.h](#).

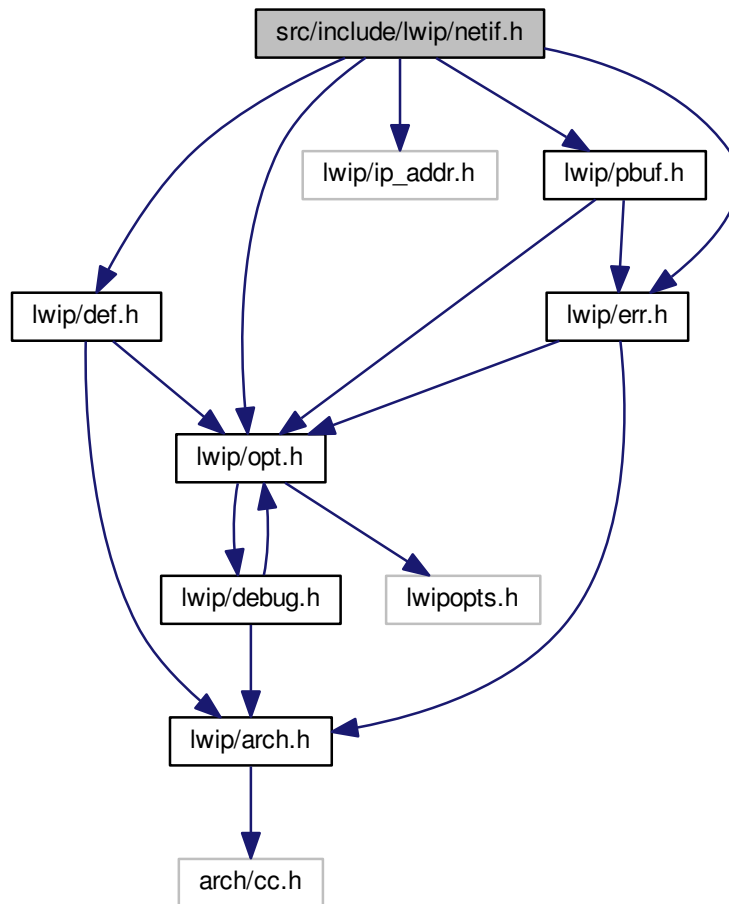
5.70 src/include/lwip/netif.h File Reference

```

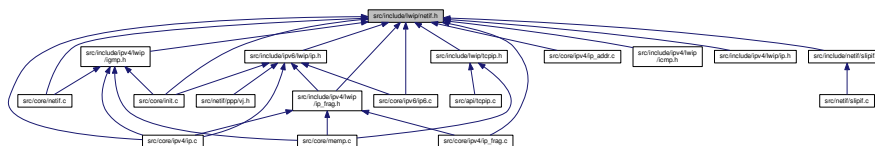
#include "lwip/opt.h"
#include "lwip/err.h"
#include "lwip/ip_addr.h"
#include "lwip/def.h"
#include "lwip/pbuf.h"

```

Include dependency graph for netif.h:



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



Data Structures

- struct [netif](#)

Macros

- #define `ENABLE_LOOPBACK` (`LWIP_NETIF_LOOPBACK` || `LWIP_HAVE_LOOPIF`)
- #define `NETIF_MAX_HWADDR_LEN` 6U

- `#define NETIF_FLAG_UP 0x01U`
- `#define NETIF_FLAG_BROADCAST 0x02U`
- `#define NETIF_FLAG_POINTTOPOINT 0x04U`
- `#define NETIF_FLAG_DHCP 0x08U`
- `#define NETIF_FLAG_LINK_UP 0x10U`
- `#define NETIF_FLAG_ETHARP 0x20U`
- `#define NETIF_FLAG_ETHERNET 0x40U`
- `#define NETIF_FLAG_IGMP 0x80U`
- `#define NETIF_INIT_SNMP(netif, type, speed)`
- `#define netif_is_up(netif) (((netif)->flags & NETIF_FLAG_UP) ? (u8_t)1 : (u8_t)0)`
- `#define netif_is_link_up(netif) (((netif)->flags & NETIF_FLAG_LINK_UP) ? (u8_t)1 : (u8_t)0)`
- `#define NETIF_SET_HWADDRHINT(netif, hint)`

Typedefs

- `typedef err_t(* netif_init_fn) (struct netif *netif)`
- `typedef err_t(* netif_input_fn) (struct pbuf *p, struct netif *inp)`
- `typedef err_t(* netif_output_fn) (struct netif *netif, struct pbuf *p, ip_addr_t *ipaddr)`
- `typedef err_t(* netif_linkoutput_fn) (struct netif *netif, struct pbuf *p)`
- `typedef void(* netif_status_callback_fn) (struct netif *netif)`
- `typedef err_t(* netif_igmp_mac_filter_fn) (struct netif *netif, ip_addr_t *group, u8_t action)`

Functions

- `void netif_init (void)`
- `struct netif * netif_add (struct netif *netif, ip_addr_t *ipaddr, ip_addr_t *netmask, ip_addr_t *gw, void *state, netif_init_fn init, netif_input_fn input)`
- `void netif_set_addr (struct netif *netif, ip_addr_t *ipaddr, ip_addr_t *netmask, ip_addr_t *gw)`
- `void netif_remove (struct netif *netif)`
- `struct netif * netif_find (char *name)`
- `void netif_set_default (struct netif *netif)`
- `void netif_set_ipaddr (struct netif *netif, ip_addr_t *ipaddr)`
- `void netif_set_netmask (struct netif *netif, ip_addr_t *netmask)`
- `void netif_set_gw (struct netif *netif, ip_addr_t *gw)`
- `void netif_set_up (struct netif *netif)`
- `void netif_set_down (struct netif *netif)`
- `void netif_set_link_up (struct netif *netif)`
- `void netif_set_link_down (struct netif *netif)`

Variables

- `struct netif * netif_list`
- `struct netif * netif_default`

5.70.1 Macro Definition Documentation

5.70.1.1 `#define ENABLE_LOOPBACK (LWIP_NETIF_LOOPBACK || LWIP_HAVE_LOOPIF)`

Definition at line 37 of file netif.h.

5.70.1.2 `#define NETIF_FLAG_BROADCAST 0x02U`

If set, the netif has broadcast capability. Set by the netif driver in its init function.

Definition at line 72 of file netif.h.

5.70.1.3 `#define NETIF_FLAG_DHCP 0x08U`

If set, the interface is configured using DHCP. Set by the DHCP code when starting or stopping DHCP.

Definition at line 78 of file netif.h.

5.70.1.4 `#define NETIF_FLAG_ETHARP 0x20U`

If set, the netif is an ethernet device using ARP. Set by the netif driver in its init function. Used to check input packet types and use of DHCP.

Definition at line 88 of file netif.h.

5.70.1.5 `#define NETIF_FLAG_ETHERNET 0x40U`

If set, the netif is an ethernet device. It might not use ARP or TCP/IP if it is used for PPPoE only.

Definition at line 92 of file netif.h.

5.70.1.6 `#define NETIF_FLAG_IGMP 0x80U`

If set, the netif has IGMP capability. Set by the netif driver in its init function.

Definition at line 95 of file netif.h.

5.70.1.7 `#define NETIF_FLAG_LINK_UP 0x10U`

If set, the interface has an active link (set by the network interface driver). Either set by the netif driver in its init function (if the link is up at that time) or at a later point once the link comes up (if link detection is supported by the hardware).

Definition at line 84 of file netif.h.

5.70.1.8 `#define NETIF_FLAG_POINTTOPOINT 0x04U`

If set, the netif is one end of a point-to-point connection. Set by the netif driver in its init function.

Definition at line 75 of file netif.h.

5.70.1.9 `#define NETIF_FLAG_UP 0x01U`

Whether the network interface is 'up'. This is a software flag used to control whether this network interface is enabled and processes traffic. It is set by the startup code (for static IP configuration) or by dhcp/autoip when an address has been assigned.

Definition at line 69 of file netif.h.

5.70.1.10 `#define NETIF_INIT_SNMP(netif, type, speed)`

Definition at line 248 of file netif.h.

5.70.1.11 **#define netif_is_link_up(netif) (((netif)->flags & NETIF_FLAG_LINK_UP) ? (u8_t)1 : (u8_t)0)**

Ask if a link is up

Definition at line 294 of file netif.h.

5.70.1.12 **#define netif_is_up(netif) (((netif)->flags & NETIF_FLAG_UP) ? (u8_t)1 : (u8_t)0)**

Ask if an interface is up

Definition at line 282 of file netif.h.

5.70.1.13 **#define NETIF_MAX_HWADDR_LEN 6U**

must be the maximum of all used hardware address lengths across all types of interfaces in use

Definition at line 61 of file netif.h.

5.70.1.14 **#define NETIF_SET_HWADDRHINT(netif, hint)**

Definition at line 321 of file netif.h.

5.70.2 Typedef Documentation

5.70.2.1 **typedef err_t(* netif_igmp_mac_filter_fn)(struct netif *netif, ip_addr_t *group, u8_t action)**

Function prototype for netif_igmp_mac_filter functions

Definition at line 130 of file netif.h.

5.70.2.2 **typedef err_t(* netif_init_fn)(struct netif *netif)**

Function prototype for netif init functions. Set up flags and output/linkoutput callback functions in this function.

Parameters

<i>netif</i>	The netif to initialize
--------------	-------------------------

Definition at line 102 of file netif.h.

5.70.2.3 **typedef err_t(* netif_input_fn)(struct pbuf *p, struct netif *inp)**

Function prototype for netif->input functions. This function is saved as 'input' callback function in the netif struct. Call it when a packet has been received.

Parameters

<i>p</i>	The received packet, copied into a pbuf
<i>inp</i>	The netif which received the packet

Definition at line 109 of file netif.h.

5.70.2.4 **typedef err_t(* netif_linkoutput_fn)(struct netif *netif, struct pbuf *p)**

Function prototype for netif->linkoutput functions. Only used for ethernet netifs. This function is called by ARP when a packet shall be sent.

Parameters

<i>netif</i>	The netif which shall send a packet
<i>p</i>	The packet to send (raw ethernet packet)

Definition at line 126 of file netif.h.

5.70.2.5 typedef err_t(* netif_output_fn) (struct netif *netif, struct pbuf *p, ip_addr_t *ipaddr)

Function prototype for netif->output functions. Called by lwIP when a packet shall be sent. For ethernet netif, set this to 'etharp_output' and set 'linkoutput'.

Parameters

<i>netif</i>	The netif which shall send a packet
<i>p</i>	The packet to send (p->payload points to IP header)
<i>ipaddr</i>	The IP address to which the packet shall be sent

Definition at line 118 of file netif.h.

5.70.2.6 typedef void(* netif_status_callback_fn) (struct netif *netif)

Function prototype for netif status- or link-callback functions.

Definition at line 128 of file netif.h.

5.70.3 Function Documentation

5.70.3.1 struct netif* netif_add (struct netif * netif, ip_addr_t * ipaddr, ip_addr_t * netmask, ip_addr_t * gw, void * state, netif_init_fn init, netif_input_fn input)

Add a network interface to the list of lwIP netifs.

Parameters

<i>netif</i>	a pre-allocated netif structure
<i>ipaddr</i>	IP address for the new netif
<i>netmask</i>	network mask for the new netif
<i>gw</i>	default gateway IP address for the new netif
<i>state</i>	opaque data passed to the new netif
<i>init</i>	callback function that initializes the interface
<i>input</i>	callback function that is called to pass ingress packets up in the protocol layer stack.

Returns

netif, or NULL if failed.

Definition at line 139 of file netif.c.

5.70.3.2 struct netif* netif_find (char * name)

Find a network interface by searching for its name

Parameters

<i>name</i>	the name of the netif (like netif->name) plus concatenated number in ascii representation (e.g. 'en0')
-------------	--

Definition at line 290 of file netif.c.

5.70.3.3 void netif_init (void)

Definition at line 106 of file netif.c.

5.70.3.4 void netif_remove (struct netif * netif)

Remove a network interface from the list of lwIP netifs.

Parameters

<i>netif</i>	the network interface to remove
--------------	---------------------------------

Definition at line 235 of file netif.c.

5.70.3.5 void netif_set_addr (struct netif * netif, ip_addr_t * ipaddr, ip_addr_t * netmask, ip_addr_t * gw)

Change IP address configuration for a network interface (including netmask and default gateway).

Parameters

<i>netif</i>	the network interface to change
<i>ipaddr</i>	the new IP address
<i>netmask</i>	the new netmask
<i>gw</i>	the new default gateway

Definition at line 221 of file netif.c.

5.70.3.6 void netif_set_default (struct netif * netif)

Set a network interface as the default network interface (used to output all packets for which no specific route is found)

Parameters

<i>netif</i>	the default network interface
--------------	-------------------------------

Definition at line 430 of file netif.c.

5.70.3.7 void netif_set_down (struct netif * netif)

Bring an interface down, disabling any traffic processing.

Note

: Enabling DHCP on a down interface will make it come up once configured.

See also

dhcp_start()

Definition at line 490 of file netif.c.

5.70.3.8 void netif_set_gw (struct netif * *netif*, ip_addr_t * *gw*)

Change the default gateway for a network interface

Parameters

<i>netif</i>	the network interface to change
<i>gw</i>	the new default gateway

Note

call [netif_set_addr\(\)](#) if you also want to change ip address and netmask

Definition at line 388 of file netif.c.

5.70.3.9 void netif_set_ipaddr (struct netif * *netif*, ip_addr_t * *ipaddr*)

Change the IP address of a network interface

Parameters

<i>netif</i>	the network interface to change
<i>ipaddr</i>	the new IP address

Note

call [netif_set_addr\(\)](#) if you also want to change netmask and default gateway

Definition at line 323 of file netif.c.

5.70.3.10 void netif_set_link_down (struct netif * *netif*)

Called by a driver when its link goes down

Definition at line 574 of file netif.c.

5.70.3.11 void netif_set_link_up (struct netif * *netif*)

Called by a driver when its link goes up

Definition at line 535 of file netif.c.

5.70.3.12 void netif_set_netmask (struct netif * *netif*, ip_addr_t * *netmask*)

Change the netmask of a network interface

Parameters

<i>netif</i>	the network interface to change
<i>netmask</i>	the new netmask

Note

call [netif_set_addr\(\)](#) if you also want to change ip address and default gateway

Definition at line 409 of file netif.c.

5.70.3.13 void netif_set_up (struct netif * *netif*)

Bring an interface up, available for processing traffic.

Note

: Enabling DHCP on a down interface will make it come up once configured.

See also

dhcp_start()

Definition at line 453 of file netif.c.

5.70.4 Variable Documentation

5.70.4.1 struct netif* netif_default

The default network interface.

Definition at line 76 of file netif.c.

5.70.4.2 struct netif* netif_list

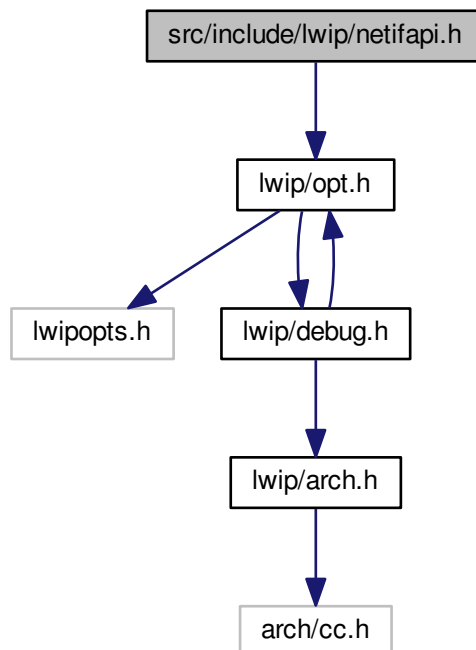
The list of network interfaces.

Definition at line 75 of file netif.c.

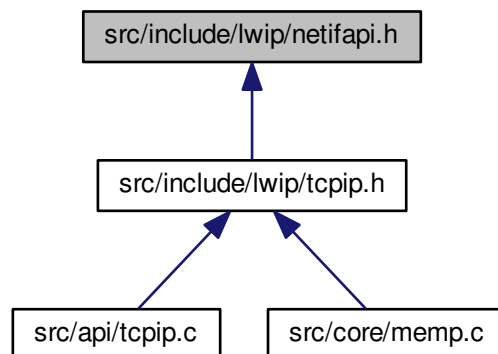
5.71 src/include/lwip/netifapi.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for netifapi.h:

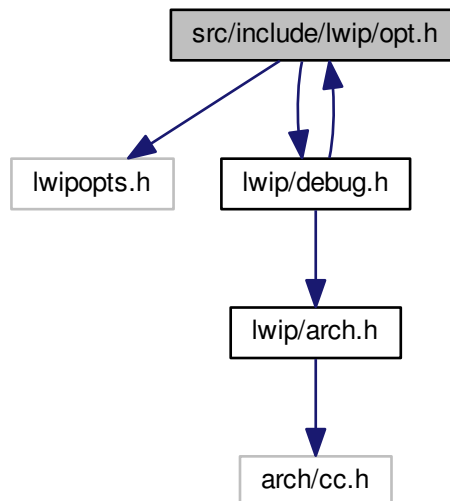


This graph shows which files directly or indirectly include this file:



5.72 src/include/lwip/opt.h File Reference

```
#include "lwipopts.h"  
#include "lwip/debug.h"  
Include dependency graph for opt.h:
```



Macros

- `#define SYS_LIGHTWEIGHT_PROT 0`
- `#define NO_SYS 0`

- #define NO_SYS_NO_TIMERS 0
- #define MEMCPY(dst, src, len) memcpy(dst,src,len)
- #define SMEMCPY(dst, src, len) memcpy(dst,src,len)
- #define MEM_LIBC_MALLOC 0
- #define MEMP_MEM_MALLOC 0
- #define MEM_ALIGNMENT 1
- #define MEM_SIZE 1600
- #define MEMP_SEPARATE_POOLS 0
- #define MEMP_OVERFLOW_CHECK 0
- #define MEMP_SANITY_CHECK 0
- #define MEM_USE_POOLS 0
- #define MEM_USE_POOLS_TRY_BIGGER_POOL 0
- #define MEMP_USE_CUSTOM_POOLS 0
- #define LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT 0
- #define MEMP_NUM_PBUF 16
- #define MEMP_NUM_RAW_PCB 4
- #define MEMP_NUM_UDP_PCB 4
- #define MEMP_NUM_TCP_PCB 5
- #define MEMP_NUM_TCP_PCB_LISTEN 8
- #define MEMP_NUM_TCP_SEG 16
- #define MEMP_NUM_REASSDATA 5
- #define MEMP_NUM_FRAG_PBUF 15
- #define MEMP_NUM_ARP_QUEUE 30
- #define MEMP_NUM_IGMP_GROUP 8
- #define MEMP_NUM_SYS_TIMEOUT (LWIP_TCP + IP_REASSEMBLY + LWIP_ARP + (2*LWIP_DHCP) + LWIP_AUTOIP + LWIP_IGMP + LWIP_DNS + PPP_SUPPORT)
- #define MEMP_NUM_NETBUF 2
- #define MEMP_NUM_NETCONN 4
- #define MEMP_NUM_TCPIP_MSG_API 8
- #define MEMP_NUM_TCPIP_MSG_INPKT 8
- #define MEMP_NUM_SNMP_NODE 50
- #define MEMP_NUM_SNMP_ROOTNODE 30
- #define MEMP_NUM_SNMP_VARBIND 2
- #define MEMP_NUM_SNMP_VALUE 3
- #define MEMP_NUM_NETDB 1
- #define MEMP_NUM_LOCALHOSTLIST 1
- #define MEMP_NUM_PPPOE_INTERFACES 1
- #define PBUF_POOL_SIZE 16
- #define LWIP_ARP 1
- #define ARP_TABLE_SIZE 10
- #define ARP_QUEUEING 0
- #define ETHARP_TRUST_IP_MAC 0
- #define ETHARP_SUPPORT_VLAN 0
- #define LWIP_ETHERNET (LWIP_ARP || PPPOE_SUPPORT)
- #define ETH_PAD_SIZE 0
- #define ETHARP_SUPPORT_STATIC_ENTRIES 0
- #define IP_FORWARD 0
- #define IP_OPTIONS_ALLOWED 1
- #define IP_REASSEMBLY 1
- #define IP_FRAG 1
- #define IP_REASS_MAXAGE 3
- #define IP_REASS_MAX_PBUFS 10
- #define IP_FRAG_USES_STATIC_BUF 0
- #define IP_DEFAULT_TTL 255
- #define IP_SOF_BROADCAST 0

- #define IP_SOF_BROADCAST_RECV 0
- #define IP_FORWARD_ALLOW_TX_ON_RX_NETIF 0
- #define LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS 0
- #define LWIP_ICMP 1
- #define ICMP_TTL (IP_DEFAULT_TTL)
- #define LWIP_BROADCAST_PING 0
- #define LWIP_MULTICAST_PING 0
- #define LWIP_RAW 1
- #define RAW_TTL (IP_DEFAULT_TTL)
- #define LWIP_DHCP 0
- #define DHCP_DOES_ARP_CHECK ((LWIP_DHCP) && (LWIP_ARP))
- #define LWIP_AUTOIP 0
- #define LWIP_DHCP_AUTOIP_COOP 0
- #define LWIP_DHCP_AUTOIP_COOP_TRIES 9
- #define LWIP_SNMP 0
- #define SNMP_CONCURRENT_REQUESTS 1
- #define SNMP_TRAP_DESTINATIONS 1
- #define SNMP_PRIVATE_MIB 0
- #define SNMP_SAFE_REQUESTS 1
- #define SNMP_MAX_OCTET_STRING_LEN 127
- #define SNMP_MAX_TREE_DEPTH 15
- #define SNMP_MAX_VALUE_SIZE LWIP_MAX((SNMP_MAX_OCTET_STRING_LEN)+1, sizeof(s32_t)*(SNMP_MAX_TREE_DEPTH))
- #define LWIP_IGMP 0
- #define LWIP_DNS 0
- #define DNS_TABLE_SIZE 4
- #define DNS_MAX_NAME_LENGTH 256
- #define DNS_MAX_SERVERS 2
- #define DNS_DOES_NAME_CHECK 1
- #define DNS_MSG_SIZE 512
- #define DNS_LOCAL_HOSTLIST 0
- #define DNS_LOCAL_HOSTLIST_IS_DYNAMIC 0
- #define LWIP_UDP 1
- #define LWIP_UDPLITE 0
- #define UDP_TTL (IP_DEFAULT_TTL)
- #define LWIP_NETBUF_RECVINFO 0
- #define LWIP_TCP 1
- #define TCP_TTL (IP_DEFAULT_TTL)
- #define TCP_WND (4 * TCP_MSS)
- #define TCP_MAXRTX 12
- #define TCP_SYNMAXRTX 6
- #define TCP_QUEUE_OOSEQ (LWIP_TCP)
- #define TCP_MSS 536
- #define TCP_CALCULATE_EFF_SEND_MSS 1
- #define TCP_SND_BUF (2 * TCP_MSS)
- #define TCP_SND_QUEUELEN ((4 * (TCP_SND_BUF) + (TCP_MSS - 1))/(TCP_MSS))
- #define TCP_SNDLOWAT LWIP_MIN(LWIP_MAX(((TCP_SND_BUF)/2), (2 * TCP_MSS) + 1), (TCP_SND_BUF) - 1)
- #define TCP_SNDQUEUELOWAT LWIP_MAX(((TCP_SND_QUEUELEN)/2), 5)
- #define TCP_OOSEQ_MAX_BYTES 0
- #define TCP_OOSEQ_MAX_PBUFS 0
- #define TCP_LISTEN_BACKLOG 0
- #define TCP_DEFAULT_LISTEN_BACKLOG 0xff
- #define TCP_OVERSIZE TCP_MSS
- #define LWIP_TCP_TIMESTAMPS 0

- `#define TCP_WND_UPDATE_THRESHOLD (TCP_WND / 4)`
- `#define LWIP_EVENT_API 0`
- `#define LWIP_CALLBACK_API 1`
- `#define PBUF_LINK_HLEN (14 + ETH_PAD_SIZE)`
- `#define PBUF_POOL_BUFSIZE LWIP_MEM_ALIGN_SIZE(TCP_MSS+40+PBUF_LINK_HLEN)`
- `#define LWIP_NETIF_HOSTNAME 0`
- `#define LWIP_NETIF_API 0`
- `#define LWIP_NETIF_STATUS_CALLBACK 0`
- `#define LWIP_NETIF_LINK_CALLBACK 0`
- `#define LWIP_NETIF_REMOVE_CALLBACK 0`
- `#define LWIP_NETIF_HWADDRHINT 0`
- `#define LWIP_NETIF_LOOPBACK 0`
- `#define LWIP_LOOPBACK_MAX_PBUFS 0`
- `#define LWIP_NETIF_LOOPBACK_MULTITHREADING (!NO_SYS)`
- `#define LWIP_NETIF_TX_SINGLE_PBUF 0`
- `#define LWIP_HAVE_LOOPIF 0`
- `#define LWIP_HAVE_SLIPIF 0`
- `#define TCPIP_THREAD_NAME "tcpip_thread"`
- `#define TCPIP_THREAD_STACKSIZE 0`
- `#define TCPIP_THREAD_PRIO 1`
- `#define TCPIP_MBOX_SIZE 0`
- `#define SLIPIF_THREAD_NAME "slipif_loop"`
- `#define SLIPIF_THREAD_STACKSIZE 0`
- `#define SLIPIF_THREAD_PRIO 1`
- `#define PPP_THREAD_NAME "pppInputThread"`
- `#define PPP_THREAD_STACKSIZE 0`
- `#define PPP_THREAD_PRIO 1`
- `#define DEFAULT_THREAD_NAME "lwIP"`
- `#define DEFAULT_THREAD_STACKSIZE 0`
- `#define DEFAULT_THREAD_PRIO 1`
- `#define DEFAULT_RAW_RECVMBOX_SIZE 0`
- `#define DEFAULT_UDP_RECVMBOX_SIZE 0`
- `#define DEFAULT_TCP_RECVMBOX_SIZE 0`
- `#define DEFAULT_ACCEPTMBOX_SIZE 0`
- `#define LWIP_TCPIP_CORE_LOCKING 0`
- `#define LWIP_TCPIP_CORE_LOCKING_INPUT 0`
- `#define LWIP_NETCONN 1`
- `#define LWIP_TCPIP_TIMEOUT 1`
- `#define LWIP_SOCKET 1`
- `#define LWIP_COMPAT_SOCKETS 1`
- `#define LWIP_POSIX_SOCKETS_IO_NAMES 1`
- `#define LWIP_TCP_KEEPALIVE 0`
- `#define LWIP_SO_SNDTIMEO 0`
- `#define LWIP_SO_RCVTIMEO 0`
- `#define LWIP_SO_RCVBUF 0`
- `#define RECV_BUFSIZE_DEFAULT INT_MAX`
- `#define SO_REUSE 0`
- `#define SO_REUSE_RXTOALL 0`
- `#define LWIP_STATS 1`
- `#define LWIP_STATS_DISPLAY 0`
- `#define LINK_STATS 1`
- `#define ETHARP_STATS (LWIP_ARP)`
- `#define IP_STATS 1`
- `#define IPFRAG_STATS (IP_REASSEMBLY || IP_FRAG)`
- `#define ICMP_STATS 1`

- `#define IGMP_STATS (LWIP_IGMP)`
- `#define UDP_STATS (LWIP_UDP)`
- `#define TCP_STATS (LWIP_TCP)`
- `#define MEM_STATS ((MEM_LIBC_MALLOC == 0) && (MEM_USE_POOLS == 0))`
- `#define MEMP_STATS (MEMP_MEM_MALLOC == 0)`
- `#define SYS_STATS (NO_SYS == 0)`
- `#define PPP_SUPPORT 0`
- `#define PPPOE_SUPPORT 0`
- `#define PPPOS_SUPPORT PPP_SUPPORT`
- `#define CHECKSUM_GEN_IP 1`
- `#define CHECKSUM_GEN_UDP 1`
- `#define CHECKSUM_GEN_TCP 1`
- `#define CHECKSUM_GEN_ICMP 1`
- `#define CHECKSUM_CHECK_IP 1`
- `#define CHECKSUM_CHECK_UDP 1`
- `#define CHECKSUM_CHECK_TCP 1`
- `#define LWIP_CHECKSUM_ON_COPY 0`
- `#define LWIP_DBG_MIN_LEVEL LWIP_DBG_LEVEL_ALL`
- `#define LWIP_DBG_TYPES_ON LWIP_DBG_ON`
- `#define ETHARP_DEBUG LWIP_DBG_OFF`
- `#define NETIF_DEBUG LWIP_DBG_OFF`
- `#define PBUF_DEBUG LWIP_DBG_OFF`
- `#define API_LIB_DEBUG LWIP_DBG_OFF`
- `#define API_MSG_DEBUG LWIP_DBG_OFF`
- `#define SOCKETS_DEBUG LWIP_DBG_OFF`
- `#define ICMP_DEBUG LWIP_DBG_OFF`
- `#define IGMP_DEBUG LWIP_DBG_OFF`
- `#define INET_DEBUG LWIP_DBG_OFF`
- `#define IP_DEBUG LWIP_DBG_OFF`
- `#define IP_REASS_DEBUG LWIP_DBG_OFF`
- `#define RAW_DEBUG LWIP_DBG_OFF`
- `#define MEM_DEBUG LWIP_DBG_OFF`
- `#define MEMP_DEBUG LWIP_DBG_OFF`
- `#define SYS_DEBUG LWIP_DBG_OFF`
- `#define TIMERS_DEBUG LWIP_DBG_OFF`
- `#define TCP_DEBUG LWIP_DBG_OFF`
- `#define TCP_INPUT_DEBUG LWIP_DBG_OFF`
- `#define TCP_FR_DEBUG LWIP_DBG_OFF`
- `#define TCP_RTO_DEBUG LWIP_DBG_OFF`
- `#define TCP_CWND_DEBUG LWIP_DBG_OFF`
- `#define TCP_WND_DEBUG LWIP_DBG_OFF`
- `#define TCP_OUTPUT_DEBUG LWIP_DBG_OFF`
- `#define TCP_RST_DEBUG LWIP_DBG_OFF`
- `#define TCP_QLEN_DEBUG LWIP_DBG_OFF`
- `#define UDP_DEBUG LWIP_DBG_OFF`
- `#define TCPIP_DEBUG LWIP_DBG_OFF`
- `#define PPP_DEBUG LWIP_DBG_OFF`
- `#define SLIP_DEBUG LWIP_DBG_OFF`
- `#define DHCP_DEBUG LWIP_DBG_OFF`
- `#define AUTOIP_DEBUG LWIP_DBG_OFF`
- `#define SNMP_MSG_DEBUG LWIP_DBG_OFF`
- `#define SNMP_MIB_DEBUG LWIP_DBG_OFF`
- `#define DNS_DEBUG LWIP_DBG_OFF`

5.72.1 Detailed Description

LwIP Options Configuration

5.72.2 Macro Definition Documentation

5.72.2.1 `#define API_LIB_DEBUG LWIP_DBG_OFF`

API_LIB_DEBUG: Enable debugging in [api_lib.c](#).

Definition at line 1919 of file opt.h.

5.72.2.2 `#define API_MSG_DEBUG LWIP_DBG_OFF`

API_MSG_DEBUG: Enable debugging in [api_msg.c](#).

Definition at line 1926 of file opt.h.

5.72.2.3 `#define ARP_QUEUEING 0`

ARP_QUEUEING==1: Multiple outgoing packets are queued during hardware address resolution. By default, only the most recent packet is queued per IP address. This is sufficient for most protocols and mainly reduces TCP connection startup time. Set this to 1 if you know your application sends more than one packet in a row to an IP address that is not in the ARP cache.

Definition at line 444 of file opt.h.

5.72.2.4 `#define ARP_TABLE_SIZE 10`

ARP_TABLE_SIZE: Number of active MAC-IP address pairs cached.

Definition at line 433 of file opt.h.

5.72.2.5 `#define AUTOIP_DEBUG LWIP_DBG_OFF`

AUTOIP_DEBUG: Enable debugging in [autoip.c](#).

Definition at line 2109 of file opt.h.

5.72.2.6 `#define CHECKSUM_CHECK_IP 1`

CHECKSUM_CHECK_IP==1: Check checksums in software for incoming IP packets.

Definition at line 1819 of file opt.h.

5.72.2.7 `#define CHECKSUM_CHECK_TCP 1`

CHECKSUM_CHECK_TCP==1: Check checksums in software for incoming TCP packets.

Definition at line 1833 of file opt.h.

5.72.2.8 `#define CHECKSUM_CHECK_UDP 1`

CHECKSUM_CHECK_UDP==1: Check checksums in software for incoming UDP packets.

Definition at line 1826 of file opt.h.

5.72.2.9 #define CHECKSUM_GEN_ICMP 1

CHECKSUM_GEN_ICMP==1: Generate checksums in software for outgoing ICMP packets.

Definition at line 1812 of file opt.h.

5.72.2.10 #define CHECKSUM_GEN_IP 1

CHECKSUM_GEN_IP==1: Generate checksums in software for outgoing IP packets.

Definition at line 1791 of file opt.h.

5.72.2.11 #define CHECKSUM_GEN_TCP 1

CHECKSUM_GEN_TCP==1: Generate checksums in software for outgoing TCP packets.

Definition at line 1805 of file opt.h.

5.72.2.12 #define CHECKSUM_GEN_UDP 1

CHECKSUM_GEN_UDP==1: Generate checksums in software for outgoing UDP packets.

Definition at line 1798 of file opt.h.

5.72.2.13 #define DEFAULT_ACCEPTMBOX_SIZE 0

DEFAULT_ACCEPTMBOX_SIZE: The mailbox size for the incoming connections. The queue size value itself is platform-dependent, but is passed to [sys_mbox_new\(\)](#) when the acceptmbox is created.

Definition at line 1379 of file opt.h.

5.72.2.14 #define DEFAULT_RAW_RECVMBOX_SIZE 0

DEFAULT_RAW_RECVMBOX_SIZE: The mailbox size for the incoming packets on a NETCONN_RAW. The queue size value itself is platform-dependent, but is passed to [sys_mbox_new\(\)](#) when the recvmbox is created.

Definition at line 1352 of file opt.h.

5.72.2.15 #define DEFAULT_TCP_RECVMBOX_SIZE 0

DEFAULT_TCP_RECVMBOX_SIZE: The mailbox size for the incoming packets on a NETCONN_TCP. The queue size value itself is platform-dependent, but is passed to [sys_mbox_new\(\)](#) when the recvmbox is created.

Definition at line 1370 of file opt.h.

5.72.2.16 #define DEFAULT_THREAD_NAME "lwIP"

DEFAULT_THREAD_NAME: The name assigned to any other lwIP thread.

Definition at line 1325 of file opt.h.

5.72.2.17 #define DEFAULT_THREAD_PRIO 1

DEFAULT_THREAD_PRIO: The priority assigned to any other lwIP thread. The priority value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1343 of file opt.h.

5.72.2.18 #define DEFAULT_THREAD_STACKSIZE 0

DEFAULT_THREAD_STACKSIZE: The stack size used by any other lwIP thread. The stack size value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1334 of file opt.h.

5.72.2.19 #define DEFAULT_UDP_RECVMBOX_SIZE 0

DEFAULT_UDP_RECVMBOX_SIZE: The mailbox size for the incoming packets on a NETCONN_UDP. The queue size value itself is platform-dependent, but is passed to [sys_mbox_new\(\)](#) when the recvmbox is created.

Definition at line 1361 of file opt.h.

5.72.2.20 #define DHCP_DEBUG LWIP_DBG_OFF

DHCP_DEBUG: Enable debugging in [dhcp.c](#).

Definition at line 2102 of file opt.h.

5.72.2.21 #define DHCP_DOES_ARP_CHECK ((LWIP_DHCP) && (LWIP_ARP))

DHCP_DOES_ARP_CHECK==1: Do an ARP check on the offered address.

Definition at line 689 of file opt.h.

5.72.2.22 #define DNS_DEBUG LWIP_DBG_OFF

DNS_DEBUG: Enable debugging for DNS.

Definition at line 2130 of file opt.h.

5.72.2.23 #define DNS_DOES_NAME_CHECK 1

DNS do a name checking between the query and the response.

Definition at line 838 of file opt.h.

5.72.2.24 #define DNS_LOCAL_HOSTLIST 0

DNS_LOCAL_HOSTLIST: Implements a local host-to-address list. If enabled, you have to define `#define DNS_LOCAL_HOSTLIST_INIT {{"host1", 0x123}, {"host2", 0x234}}` (an array of structs name/address, where address is an `u32_t` in network byte order).

Instead, you can also use an external function: `#define DNS_LOOKUP_LOCAL_EXTERN(x) extern u32_t my_lookup_function(const char *name)` that returns the IP address or `INADDR_NONE` if not found.

Definition at line 857 of file opt.h.

5.72.2.25 #define DNS_LOCAL_HOSTLIST_IS_DYNAMIC 0

If this is turned on, the local host-list can be dynamically changed at runtime.

Definition at line 863 of file opt.h.

5.72.2.26 #define DNS_MAX_NAME_LENGTH 256

DNS maximum host name length supported in the name table.

Definition at line 828 of file opt.h.

5.72.2.27 #define DNS_MAX_SERVERS 2

The maximum of DNS servers

Definition at line 833 of file opt.h.

5.72.2.28 #define DNS_MSG_SIZE 512

DNS message max. size. Default value is RFC compliant.

Definition at line 843 of file opt.h.

5.72.2.29 #define DNS_TABLE_SIZE 4

DNS maximum number of entries to maintain locally.

Definition at line 823 of file opt.h.

5.72.2.30 #define ETH_PAD_SIZE 0

ETH_PAD_SIZE: number of bytes added before the ethernet header to ensure alignment of payload after that header. Since the header is 14 bytes long, without this padding e.g. addresses in the IP header will not be aligned on a 32-bit boundary, so setting this to 2 can speed up 32-bit-platforms.

Definition at line 486 of file opt.h.

5.72.2.31 #define ETHARP_DEBUG LWIP_DBG_OFF

ETHARP_DEBUG: Enable debugging in [etharp.c](#).

Definition at line 1898 of file opt.h.

5.72.2.32 #define ETHARP_STATS (LWIP_ARP)

ETHARP_STATS==1: Enable etharp stats.

Definition at line 1533 of file opt.h.

5.72.2.33 #define ETHARP_SUPPORT_STATIC_ENTRIES 0

ETHARP_SUPPORT_STATIC_ENTRIES==1: enable code to support static ARP table entries (using [etharp_add_static_entry](#)/[etharp_remove_static_entry](#)).

Definition at line 493 of file opt.h.

5.72.2.34 #define ETHARP_SUPPORT_VLAN 0

ETHARP_SUPPORT_VLAN==1: support receiving ethernet packets with VLAN header. Additionally, you can define ETHARP_VLAN_CHECK to an `u16_t` VLAN ID to check. If ETHARP_VLAN_CHECK is defined, only VLAN-N-traffic for this VLAN is accepted. If ETHARP_VLAN_CHECK is not defined, all traffic is accepted. Alternatively,

define a function/define ETHARP_VLAN_CHECK_FN(eth_hdr, vlan) that returns 1 to accept a packet or 0 to drop a packet.

Definition at line 470 of file opt.h.

5.72.2.35 #define ETHARP_TRUST_IP_MAC 0

ETHARP_TRUST_IP_MAC==1: Incoming IP packets cause the ARP table to be updated with the source MAC and IP addresses supplied in the packet. You may want to disable this if you do not trust LAN peers to have the correct addresses, or as a limited approach to attempt to handle spoofing. If disabled, lwIP will need to make a new ARP request if the peer is not already in the ARP table, adding a little latency. The peer *is* in the ARP table if it requested our address before. Also notice that this slows down input processing of every IP packet!

Definition at line 458 of file opt.h.

5.72.2.36 #define ICMP_DEBUG LWIP_DBG_OFF

ICMP_DEBUG: Enable debugging in [icmp.c](#).

Definition at line 1940 of file opt.h.

5.72.2.37 #define ICMP_STATS 1

ICMP_STATS==1: Enable ICMP stats.

Definition at line 1555 of file opt.h.

5.72.2.38 #define ICMP_TTL (IP_DEFAULT_TTL)

ICMP_TTL: Default value for Time-To-Live used by ICMP packets.

Definition at line 637 of file opt.h.

5.72.2.39 #define IGMP_DEBUG LWIP_DBG_OFF

IGMP_DEBUG: Enable debugging in [igmp.c](#).

Definition at line 1947 of file opt.h.

5.72.2.40 #define IGMP_STATS (LWIP_IGMP)

IGMP_STATS==1: Enable IGMP stats.

Definition at line 1562 of file opt.h.

5.72.2.41 #define INET_DEBUG LWIP_DBG_OFF

INET_DEBUG: Enable debugging in [inet.c](#).

Definition at line 1954 of file opt.h.

5.72.2.42 #define IP_DEBUG LWIP_DBG_OFF

IP_DEBUG: Enable debugging for IP.

Definition at line 1961 of file opt.h.

5.72.2.43 `#define IP_DEFAULT_TTL 255`

`IP_FRAG_MAX_MTU`: Assumed max MTU on any interface for IP frag buffer (requires `IP_FRAG_USES_STATIC_BUF==1`) `IP_DEFAULT_TTL`: Default value for Time-To-Live used by transport layers.

Definition at line 580 of file `opt.h`.

5.72.2.44 `#define IP_FORWARD 0`

`IP_FORWARD==1`: Enables the ability to forward IP packets across network interfaces. If you are going to run lwIP on a device with only one network interface, define this to 0.

Definition at line 508 of file `opt.h`.

5.72.2.45 `#define IP_FORWARD_ALLOW_TX_ON_RX_NETIF 0`

`IP_FORWARD_ALLOW_TX_ON_RX_NETIF==1`: allow `ip_forward()` to send packets back out on the netif where it was received. This should only be used for wireless networks. ATTENTION: When this is 1, make sure your netif driver correctly marks incoming link-layer-broadcast/multicast packets as such using the corresponding pbuf flags!

Definition at line 608 of file `opt.h`.

5.72.2.46 `#define IP_FRAG 1`

`IP_FRAG==1`: Fragment outgoing IP packets if their size exceeds MTU. Note that this option does not affect incoming packet sizes, which can be controlled via `IP_REASSEMBLY`.

Definition at line 535 of file `opt.h`.

5.72.2.47 `#define IP_FRAG_USES_STATIC_BUF 0`

`IP_FRAG_USES_STATIC_BUF==1`: Use a static MTU-sized buffer for IP fragmentation. Otherwise pbufs are allocated and reference the original packet data to be fragmented (or with `LWIP_NETIF_TX_SINGLE_PBUF==1`, new `PBUF_RAM` pbufs are used for fragments). ATTENTION: `IP_FRAG_USES_STATIC_BUF==1` may not be used for DMA-enabled MACs!

Definition at line 565 of file `opt.h`.

5.72.2.48 `#define IP_OPTIONS_ALLOWED 1`

`IP_OPTIONS_ALLOWED`: Defines the behavior for IP options. `IP_OPTIONS_ALLOWED==0`: All packets with IP options are dropped. `IP_OPTIONS_ALLOWED==1`: IP options are allowed (but not parsed).

Definition at line 517 of file `opt.h`.

5.72.2.49 `#define IP_REASS_DEBUG LWIP_DBG_OFF`

`IP_REASS_DEBUG`: Enable debugging in [ip_frag.c](#) for both frag & reass.

Definition at line 1968 of file `opt.h`.

5.72.2.50 `#define IP_REASS_MAX_PBUFS 10`

`IP_REASS_MAX_PBUFS`: Total maximum amount of pbufs waiting to be reassembled. Since the received pbufs are enqueued, be sure to configure `PBUF_POOL_SIZE > IP_REASS_MAX_PBUFS` so that the stack is still able to receive packets even if the maximum amount of fragments is enqueued for reassembly!

Definition at line 554 of file opt.h.

5.72.2.51 `#define IP_REASS_MAXAGE 3`

IP_REASS_MAXAGE: Maximum time (in multiples of IP_TMR_INTERVAL - so seconds, normally) a fragmented IP packet waits for all fragments to arrive. If not all fragments arrived in this time, the whole packet is discarded.

Definition at line 544 of file opt.h.

5.72.2.52 `#define IP_REASSEMBLY 1`

IP_REASSEMBLY==1: Reassemble incoming fragmented IP packets. Note that this option does not affect outgoing packet sizes, which can be controlled via IP_FRAG.

Definition at line 526 of file opt.h.

5.72.2.53 `#define IP_SOF_BROADCAST 0`

IP_SOF_BROADCAST=1: Use the SOF_BROADCAST field to enable broadcast filter per pcb on udp and raw send operations. To enable broadcast filter on recv operations, you also have to set IP_SOF_BROADCAST_RECV=1.

Definition at line 589 of file opt.h.

5.72.2.54 `#define IP_SOF_BROADCAST_RECV 0`

IP_SOF_BROADCAST_RECV (requires IP_SOF_BROADCAST=1) enable the broadcast filter on recv operations.

Definition at line 597 of file opt.h.

5.72.2.55 `#define IP_STATS 1`

IP_STATS==1: Enable IP stats.

Definition at line 1540 of file opt.h.

5.72.2.56 `#define IPFRAG_STATS (IP_REASSEMBLY || IP_FRAG)`

IPFRAG_STATS==1: Enable IP fragmentation stats. Default is on if using either frag or reass.

Definition at line 1548 of file opt.h.

5.72.2.57 `#define LINK_STATS 1`

LINK_STATS==1: Enable link stats.

Definition at line 1526 of file opt.h.

5.72.2.58 `#define LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT 0`

Set this to 1 if you want to free PBUF_RAM pbufs (or call [mem_free\(\)](#)) from interrupt context (or another context that doesn't allow waiting for a semaphore). If set to 1, mem_malloc will be protected by a semaphore and SYS_ARCH_PROTECT, while mem_free will only use SYS_ARCH_PROTECT. mem_malloc SYS_ARCH_UNPROTECTs with each loop so that mem_free can run.

ATTENTION: As you can see from the above description, this leads to dis-/ enabling interrupts often, which can be slow! Also, on low memory, mem_malloc can need longer.

If you don't want that, at least for `NO_SYS=0`, you can still use the following functions to enqueue a deallocation call which then runs in the `tcpip_thread` context:

- `pbuf_free_callback(p);`
- `mem_free_callback(m);`

Definition at line 212 of file `opt.h`.

5.72.2.59 `#define LWIP_ARP 1`

`LWIP_ARP==1`: Enable ARP functionality.

Definition at line 426 of file `opt.h`.

5.72.2.60 `#define LWIP_AUTOIP 0`

`LWIP_AUTOIP==1`: Enable AUTOIP module.

Definition at line 701 of file `opt.h`.

5.72.2.61 `#define LWIP_BROADCAST_PING 0`

`LWIP_BROADCAST_PING==1`: respond to broadcast pings (default is unicast only)

Definition at line 644 of file `opt.h`.

5.72.2.62 `#define LWIP_CALLBACK_API 1`

Definition at line 1080 of file `opt.h`.

5.72.2.63 `#define LWIP_CHECKSUM_ON_COPY 0`

`LWIP_CHECKSUM_ON_COPY==1`: Calculate checksum when copying data from application buffers to pbufs.

Definition at line 1841 of file `opt.h`.

5.72.2.64 `#define LWIP_COMPAT_SOCKETS 1`

`LWIP_COMPAT_SOCKETS==1`: Enable BSD-style sockets functions names. (only used if you use [sockets.c](#))

Definition at line 1434 of file `opt.h`.

5.72.2.65 `#define LWIP_DBG_MIN_LEVEL LWIP_DBG_LEVEL_ALL`

`LWIP_HOOK_IP4_INPUT(pbuf, input_netif)`:

- called from [ip_input\(\)](#) (IPv4)
- `pbuf`: received struct pbuf passed to [ip_input\(\)](#)
- `input_netif`: struct netif on which the packet has been received Return values:
- 0: Hook has not consumed the packet, packet is processed as normal
- != 0: Hook has consumed the packet. If the hook consumed the packet, 'pbuf' is in the responsibility of the hook (i.e. free it when done). `LWIP_HOOK_IP4_ROUTE(dest)`:

- called from `ip_route()` (IPv4)
- `dest`: destination IPv4 address Returns the destination netif or NULL if no destination netif is found. In that case, `ip_route()` continues as normal. `LWIP_DBG_MIN_LEVEL`: After masking, the value of the debug is compared against this value. If it is smaller, then debugging messages are written.

Definition at line 1883 of file opt.h.

5.72.2.66 `#define LWIP_DBG_TYPES_ON LWIP_DBG_ON`

`LWIP_DBG_TYPES_ON`: A mask that can be used to globally enable/disable debug messages of certain types.

Definition at line 1891 of file opt.h.

5.72.2.67 `#define LWIP_DHCP 0`

`LWIP_DHCP==1`: Enable DHCP module.

Definition at line 682 of file opt.h.

5.72.2.68 `#define LWIP_DHCP_AUTOIP_COOP 0`

`LWIP_DHCP_AUTOIP_COOP==1`: Allow DHCP and AUTOIP to be both enabled on the same interface at the same time.

Definition at line 709 of file opt.h.

5.72.2.69 `#define LWIP_DHCP_AUTOIP_COOP_TRIES 9`

`LWIP_DHCP_AUTOIP_COOP_TRIES`: Set to the number of DHCP DISCOVER probes that should be sent before falling back on AUTOIP. This can be set as low as 1 to get an AutoIP address very quickly, but you should be prepared to handle a changing IP address when DHCP overrides AutoIP.

Definition at line 720 of file opt.h.

5.72.2.70 `#define LWIP_DNS 0`

`LWIP_DNS==1`: Turn on DNS module. UDP must be available for DNS transport.

Definition at line 818 of file opt.h.

5.72.2.71 `#define LWIP_ETHERNET (LWIP_ARP || PPPoE_SUPPORT)`

`LWIP_ETHERNET==1`: enable ethernet support for PPPoE even though ARP might be disabled

Definition at line 477 of file opt.h.

5.72.2.72 `#define LWIP_EVENT_API 0`

`LWIP_EVENT_API` and `LWIP_CALLBACK_API`: Only one of these should be set to 1. `LWIP_EVENT_API==1`: The user defines `lwip_tcp_event()` to receive all events (accept, sent, etc) that happen in the system. `LWIP_CALLBACK_API==1`: The PCB callback function is called directly for the event. This is the default.

Definition at line 1079 of file opt.h.

5.72.2.73 #define LWIP_HAVE_LOOPIF 0

LWIP_HAVE_LOOPIF==1: Support loop interface (127.0.0.1) and loopif.c

Definition at line 1217 of file opt.h.

5.72.2.74 #define LWIP_HAVE_SLIPIF 0

LWIP_HAVE_SLIPIF==1: Support slip interface and [slipif.c](#)

Definition at line 1229 of file opt.h.

5.72.2.75 #define LWIP_ICMP 1

LWIP_ICMP==1: Enable ICMP module inside the IP stack. Be careful, disable that make your product non-compliant to RFC1122

Definition at line 630 of file opt.h.

5.72.2.76 #define LWIP_IGMP 0

LWIP_IGMP==1: Turn on IGMP module.

Definition at line 805 of file opt.h.

5.72.2.77 #define LWIP_LOOPBACK_MAX_PBUFS 0

LWIP_LOOPBACK_MAX_PBUFS: Maximum number of pbufs on queue for loopback sending for each netif (0 = disabled)

Definition at line 1175 of file opt.h.

5.72.2.78 #define LWIP_MULTICAST_PING 0

LWIP_MULTICAST_PING==1: respond to multicast pings (default is unicast only)

Definition at line 651 of file opt.h.

5.72.2.79 #define LWIP_NETBUF_RECVINFO 0

LWIP_NETBUF_RECVINFO==1: append destination addr and port to every netbuf.

Definition at line 896 of file opt.h.

5.72.2.80 #define LWIP_NETCONN 1

LWIP_NETCONN==1: Enable Netconn API (require to use [api_lib.c](#))

Definition at line 1407 of file opt.h.

5.72.2.81 #define LWIP_NETIF_API 0

LWIP_NETIF_API==1: Support netif api (in [netifapi.c](#))

Definition at line 1124 of file opt.h.

5.72.2.82 #define LWIP_NETIF_HOSTNAME 0

LWIP_NETIF_HOSTNAME==1: use DHCP_OPTION_HOSTNAME with netif's hostname field.

Definition at line 1117 of file opt.h.

5.72.2.83 #define LWIP_NETIF_HWADDRHINT 0

LWIP_NETIF_HWADDRHINT==1: Cache link-layer-address hints (e.g. table indices) in struct netif. TCP and UDP can make use of this to prevent scanning the ARP table for every sent packet. While this is faster for big ARP tables or many concurrent connections, it might be counterproductive if you have a tiny ARP table or if there never are concurrent connections.

Definition at line 1159 of file opt.h.

5.72.2.84 #define LWIP_NETIF_LINK_CALLBACK 0

LWIP_NETIF_LINK_CALLBACK==1: Support a callback function from an interface whenever the link changes (i.e., link down)

Definition at line 1140 of file opt.h.

5.72.2.85 #define LWIP_NETIF_LOOPBACK 0

LWIP_NETIF_LOOPBACK==1: Support sending packets with a destination IP address equal to the netif IP address, looping them back up the stack.

Definition at line 1167 of file opt.h.

5.72.2.86 #define LWIP_NETIF_LOOPBACK_MULTITHREADING (!NO_SYS)

LWIP_NETIF_LOOPBACK_MULTITHREADING: Indicates whether threading is enabled in the system, as netifs must change how they behave depending on this setting for the LWIP_NETIF_LOOPBACK option to work. Setting this is needed to avoid reentering non-reentrant functions like tcp_input(). LWIP_NETIF_LOOPBACK_MULTITHREADING==1: Indicates that the user is using a multithreaded environment like [tcpip.c](#). In this case, netif->input() is called directly. LWIP_NETIF_LOOPBACK_MULTITHREADING==0: Indicates a polling (or NO_SYS) setup. The packets are put on a list and netif_poll() must be called in the main application loop.

Definition at line 1192 of file opt.h.

5.72.2.87 #define LWIP_NETIF_REMOVE_CALLBACK 0

LWIP_NETIF_REMOVE_CALLBACK==1: Support a callback function that is called when a netif has been removed

Definition at line 1148 of file opt.h.

5.72.2.88 #define LWIP_NETIF_STATUS_CALLBACK 0

LWIP_NETIF_STATUS_CALLBACK==1: Support a callback function whenever an interface changes its up/down status (i.e., due to DHCP IP acquisition)

Definition at line 1132 of file opt.h.

5.72.2.89 #define LWIP_NETIF_TX_SINGLE_PBUF 0

LWIP_NETIF_TX_SINGLE_PBUF: if this is set to 1, lwip tries to put all data to be sent into one single pbuf. This is for compatibility with DMA-enabled MACs that do not support scatter-gather. Beware that this might involve

CPU-memcpy before transmitting that would not be needed without this flag! Use this only if you need to!

Todo : TCP and IP-frag do not work with this, yet:

Definition at line 1205 of file opt.h.

5.72.2.90 `#define LWIP_POSIX_SOCKETS_IO_NAMES 1`

LWIP_POSIX_SOCKETS_IO_NAMES==1: Enable POSIX-style sockets functions names. Disable this option if you use a POSIX operating system that uses the same names (read, write & close). (only used if you use [sockets.c](#))

Definition at line 1443 of file opt.h.

5.72.2.91 `#define LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS 0`

LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS==1: randomize the local port for the first local TCP/UDP pcb (default==0). This can prevent creating predictable port numbers after booting a device.

Definition at line 617 of file opt.h.

5.72.2.92 `#define LWIP_RAW 1`

LWIP_RAW==1: Enable application layer to hook into the IP layer itself.

Definition at line 663 of file opt.h.

5.72.2.93 `#define LWIP_SNMP 0`

LWIP_SNMP==1: Turn on SNMP module. UDP must be available for SNMP transport.

Definition at line 733 of file opt.h.

5.72.2.94 `#define LWIP_SO_RCVBUF 0`

LWIP_SO_RCVBUF==1: Enable SO_RCVBUF processing.

Definition at line 1475 of file opt.h.

5.72.2.95 `#define LWIP_SO_RCVTIMEO 0`

LWIP_SO_RCVTIMEO==1: Enable receive timeout for sockets/netconns and SO_RCVTIMEO processing.

Definition at line 1468 of file opt.h.

5.72.2.96 `#define LWIP_SO_SNDTIMEO 0`

LWIP_SO_SNDTIMEO==1: Enable send timeout for sockets/netconns and SO_SNDTIMEO processing.

Definition at line 1460 of file opt.h.

5.72.2.97 `#define LWIP_SOCKET 1`

LWIP_SOCKET==1: Enable Socket API (require to use [sockets.c](#))

Definition at line 1426 of file opt.h.

5.72.2.98 #define LWIP_STATS 1

LWIP_STATS==1: Enable statistics collection in lwip_stats.

Definition at line 1510 of file opt.h.

5.72.2.99 #define LWIP_STATS_DISPLAY 0

LWIP_STATS_DISPLAY==1: Compile in the statistics output functions.

Definition at line 1519 of file opt.h.

5.72.2.100 #define LWIP_TCP 1

LWIP_TCP==1: Turn on TCP.

Definition at line 908 of file opt.h.

5.72.2.101 #define LWIP_TCP_KEEPALIVE 0

LWIP_TCP_KEEPALIVE==1: Enable TCP_KEEPIIDLE, TCP_KEEPINTVL and TCP_KEEPCNT options processing. Note that TCP_KEEPIIDLE and TCP_KEEPINTVL have to be set in seconds. (does not require [sockets.c](#), and will affect [tcp.c](#))

Definition at line 1452 of file opt.h.

5.72.2.102 #define LWIP_TCP_TIMESTAMPS 0

LWIP_TCP_TIMESTAMPS==1: support the TCP timestamp option.

Definition at line 1060 of file opt.h.

5.72.2.103 #define LWIP_TCPIP_CORE_LOCKING 0

LWIP_TCPIP_CORE_LOCKING: (EXPERIMENTAL!) Don't use it if you're not an active lwIP project member

Definition at line 1392 of file opt.h.

5.72.2.104 #define LWIP_TCPIP_CORE_LOCKING_INPUT 0

LWIP_TCPIP_CORE_LOCKING_INPUT: (EXPERIMENTAL!) Don't use it if you're not an active lwIP project member

Definition at line 1400 of file opt.h.

5.72.2.105 #define LWIP_TCPIP_TIMEOUT 1

LWIP_TCPIP_TIMEOUT==1: Enable tcpip_timeout/tcpip_untimeout to create timers running in tcpip_thread from another thread.

Definition at line 1414 of file opt.h.

5.72.2.106 #define LWIP_UDP 1

LWIP_UDP==1: Turn on UDP.

Definition at line 875 of file opt.h.

5.72.2.107 #define LWIP_UDPLITE 0

LWIP_UDPLITE==1: Turn on UDP-Lite. (Requires LWIP_UDP)

Definition at line 882 of file opt.h.

5.72.2.108 #define MEM_ALIGNMENT 1

MEM_ALIGNMENT: should be set to the alignment of the CPU 4 byte alignment -> #define MEM_ALIGNMENT 4
2 byte alignment -> #define MEM_ALIGNMENT 2

Definition at line 124 of file opt.h.

5.72.2.109 #define MEM_DEBUG LWIP_DBG_OFF

MEM_DEBUG: Enable debugging in [mem.c](#).

Definition at line 1982 of file opt.h.

5.72.2.110 #define MEM_LIBC_MALLOC 0

MEM_LIBC_MALLOC==1: Use malloc/free/realloc provided by your C-library instead of the lwip internal allocator.
Can save code size if you already use it.

Definition at line 106 of file opt.h.

5.72.2.111 #define MEM_SIZE 1600

MEM_SIZE: the size of the heap memory. If the application will send a lot of data that needs to be copied, this should be set high.

Definition at line 132 of file opt.h.

5.72.2.112 #define MEM_STATS ((MEM_LIBC_MALLOC == 0) && (MEM_USE_POOLS == 0))

MEM_STATS==1: Enable [mem.c](#) stats.

Definition at line 1585 of file opt.h.

5.72.2.113 #define MEM_USE_POOLS 0

MEM_USE_POOLS==1: Use an alternative to malloc() by allocating from a set of memory pools of various sizes. When mem_malloc is called, an element of the smallest pool that can provide the length needed is returned. To use this, MEM_USE_CUSTOM_POOLS also has to be enabled.

Definition at line 172 of file opt.h.

5.72.2.114 #define MEM_USE_POOLS_TRY_BIGGER_POOL 0

MEM_USE_POOLS_TRY_BIGGER_POOL==1: if one malloc-pool is empty, try the next bigger pool - WARNING: THIS MIGHT WASTE MEMORY but it can make a system more reliable.

Definition at line 180 of file opt.h.

5.72.2.115 `#define MEMCPY(dst, src, len) memcpy(dst,src,len)`

MEMCPY: override this if you have a faster implementation at hand than the one included in your C library

Definition at line 84 of file opt.h.

5.72.2.116 `#define MEMP_DEBUG LWIP_DBG_OFF`

MEMP_DEBUG: Enable debugging in [memp.c](#).

Definition at line 1989 of file opt.h.

5.72.2.117 `#define MEMP_MEM_MALLOC 0`

MEMP_MEM_MALLOC==1: Use mem_malloc/mem_free instead of the lwip pool allocator. Especially useful with MEM_LIBC_MALLOC but handle with care regarding execution speed and usage from interrupts!

Definition at line 115 of file opt.h.

5.72.2.118 `#define MEMP_NUM_ARP_QUEUE 30`

MEMP_NUM_ARP_QUEUE: the number of simultaneously queued outgoing packets (pbufs) that are waiting for an ARP request (to resolve their destination address) to finish. (requires the ARP_QUEUEING option)

Definition at line 296 of file opt.h.

5.72.2.119 `#define MEMP_NUM_FRAG_PBUF 15`

MEMP_NUM_FRAG_PBUF: the number of IP fragments simultaneously sent (fragments, not whole packets!). This is only used with IP_FRAG_USES_STATIC_BUF==0 and LWIP_NETIF_TX_SINGLE_PBUF==0 and only has to be > 1 with DMA-enabled MACs where the packet is not yet sent when netif->output returns.

Definition at line 286 of file opt.h.

5.72.2.120 `#define MEMP_NUM_IGMP_GROUP 8`

MEMP_NUM_IGMP_GROUP: The number of multicast groups whose network interfaces can be members at the same time (one per netif - allsystems group -, plus one per netif membership). (requires the LWIP_IGMP option)

Definition at line 306 of file opt.h.

5.72.2.121 `#define MEMP_NUM_LOCALHOSTLIST 1`

MEMP_NUM_LOCALHOSTLIST: the number of host entries in the local host list if DNS_LOCAL_HOSTLIST_IS_DYNAMIC==1.

Definition at line 399 of file opt.h.

5.72.2.122 `#define MEMP_NUM_NETBUF 2`

MEMP_NUM_NETBUF: the number of struct netbufs. (only needed if you use the sequential API, like [api_lib.c](#))

Definition at line 324 of file opt.h.

5.72.2.123 #define MEMP_NUM_NETCONN 4

MEMP_NUM_NETCONN: the number of struct netconns. (only needed if you use the sequential API, like [api_lib.c](#))

Definition at line 332 of file opt.h.

5.72.2.124 #define MEMP_NUM_NETDB 1

MEMP_NUM_NETDB: the number of concurrently running lwip_addrinfo() calls (before freeing the corresponding memory using lwip_freeaddrinfo()).

Definition at line 391 of file opt.h.

5.72.2.125 #define MEMP_NUM_PBUF 16

MEMP_NUM_PBUF: the number of memp struct pbufs (used for PBUF_ROM and PBUF_REF). If the application sends a lot of data out of ROM (or other static memory), this should be set high.

Definition at line 226 of file opt.h.

5.72.2.126 #define MEMP_NUM_PPPOE_INTERFACES 1

MEMP_NUM_PPPOE_INTERFACES: the number of concurrently active PPPoE interfaces (only used with PPP↔OE_SUPPORT==1)

Definition at line 407 of file opt.h.

5.72.2.127 #define MEMP_NUM_RAW_PCB 4

MEMP_NUM_RAW_PCB: Number of raw connection PCBs (requires the LWIP_RAW option)

Definition at line 234 of file opt.h.

5.72.2.128 #define MEMP_NUM_REASSDATA 5

MEMP_NUM_REASSDATA: the number of IP packets simultaneously queued for reassembly (whole packets, not fragments!)

Definition at line 275 of file opt.h.

5.72.2.129 #define MEMP_NUM_SNMP_NODE 50

MEMP_NUM_SNMP_NODE: the number of leafs in the SNMP tree.

Definition at line 357 of file opt.h.

5.72.2.130 #define MEMP_NUM_SNMP_ROOTNODE 30

MEMP_NUM_SNMP_ROOTNODE: the number of branches in the SNMP tree. Every branch has one leaf (MEM↔P_NUM_SNMP_NODE) at least!

Definition at line 365 of file opt.h.

5.72.2.131 #define MEMP_NUM_SNMP_VALUE 3

MEMP_NUM_SNMP_VALUE: the number of OID or values concurrently used (does not have to be changed normally) - 3 of these are used per request (1 for the value read and 2 for OIDs - input and output)

Definition at line 383 of file opt.h.

5.72.2.132 #define MEMP_NUM_SNMP_VARBIND 2

MEMP_NUM_SNMP_VARBIND: the number of concurrent requests (does not have to be changed normally) - 2 of these are used per request (1 for input, 1 for output)

Definition at line 374 of file opt.h.

5.72.2.133 #define MEMP_NUM_SYS_TIMEOUT (LWIP_TCP + IP_REASSEMBLY + LWIP_ARP + (2*LWIP_DHCP) + LWIP_AUTOIP + LWIP_IGMP + LWIP_DNS + PPP_SUPPORT)

MEMP_NUM_SYS_TIMEOUT: the number of simultaneously active timeouts. (requires NO_SYS==0) The default number of timeouts is calculated here for all enabled modules. The formula expects settings to be either '0' or '1'.

Definition at line 316 of file opt.h.

5.72.2.134 #define MEMP_NUM_TCP_PCB 5

MEMP_NUM_TCP_PCB: the number of simulatenously active TCP connections. (requires the LWIP_TCP option)

Definition at line 251 of file opt.h.

5.72.2.135 #define MEMP_NUM_TCP_PCB_LISTEN 8

MEMP_NUM_TCP_PCB_LISTEN: the number of listening TCP connections. (requires the LWIP_TCP option)

Definition at line 259 of file opt.h.

5.72.2.136 #define MEMP_NUM_TCP_SEG 16

MEMP_NUM_TCP_SEG: the number of simultaneously queued TCP segments. (requires the LWIP_TCP option)

Definition at line 267 of file opt.h.

5.72.2.137 #define MEMP_NUM_TCPIP_MSG_API 8

MEMP_NUM_TCPIP_MSG_API: the number of struct [tcpip_msg](#), which are used for callback/timeout API communication. (only needed if you use [tcpip.c](#))

Definition at line 341 of file opt.h.

5.72.2.138 #define MEMP_NUM_TCPIP_MSG_INPKT 8

MEMP_NUM_TCPIP_MSG_INPKT: the number of struct [tcpip_msg](#), which are used for incoming packets. (only needed if you use [tcpip.c](#))

Definition at line 350 of file opt.h.

5.72.2.139 #define MEMP_NUM_UDP_PCB 4

MEMP_NUM_UDP_PCB: the number of UDP protocol control blocks. One per active UDP "connection". (requires the LWIP_UDP option)

Definition at line 243 of file opt.h.

5.72.2.140 #define MEMP_OVERFLOW_CHECK 0

MEMP_OVERFLOW_CHECK: memp overflow protection reserves a configurable amount of bytes before and after each memp element in every pool and fills it with a prominent default value. MEMP_OVERFLOW_CHECK == 0 no checking MEMP_OVERFLOW_CHECK == 1 checks each element when it is freed MEMP_OVERFLOW_CHECK >= 2 checks each element in every pool every time [memp_malloc\(\)](#) or [memp_free\(\)](#) is called (useful but slow!)

Definition at line 154 of file opt.h.

5.72.2.141 #define MEMP_SANITY_CHECK 0

MEMP_SANITY_CHECK==1: run a sanity check after each [memp_free\(\)](#) to make sure that there are no cycles in the linked lists.

Definition at line 162 of file opt.h.

5.72.2.142 #define MEMP_SEPARATE_POOLS 0

MEMP_SEPARATE_POOLS: if defined to 1, each pool is placed in its own array. This can be used to individually change the location of each pool. Default is one big array for all pools

Definition at line 141 of file opt.h.

5.72.2.143 #define MEMP_STATS (MEMP_MEM_MALLOC == 0)

MEMP_STATS==1: Enable [memp.c](#) pool stats.

Definition at line 1592 of file opt.h.

5.72.2.144 #define MEMP_USE_CUSTOM_POOLS 0

MEMP_USE_CUSTOM_POOLS==1: whether to include a user file lwippools.h that defines additional pools beyond the "standard" ones required by lwIP. If you set this to 1, you must have lwippools.h in your include path somewhere.

Definition at line 190 of file opt.h.

5.72.2.145 #define NETIF_DEBUG LWIP_DBG_OFF

NETIF_DEBUG: Enable debugging in [netif.c](#).

Definition at line 1905 of file opt.h.

5.72.2.146 #define NO_SYS 0

NO_SYS==1: Provides VERY minimal functionality. Otherwise, use lwIP facilities.

Definition at line 68 of file opt.h.

5.72.2.147 #define NO_SYS_NO_TIMERS 0

NO_SYS_NO_TIMERS==1: Drop support for sys_timeout when NO_SYS==1. Mainly for compatibility to old versions.

Definition at line 76 of file opt.h.

5.72.2.148 #define PBUF_DEBUG LWIP_DBG_OFF

PBUF_DEBUG: Enable debugging in [pbuf.c](#).

Definition at line 1912 of file opt.h.

5.72.2.149 #define PBUF_LINK_HLEN (14 + ETH_PAD_SIZE)

PBUF_LINK_HLEN: the number of bytes that should be allocated for a link level header. The default is 14, the standard value for Ethernet.

Definition at line 1095 of file opt.h.

5.72.2.150 #define PBUF_POOL_BUFSIZE LWIP_MEM_ALIGN_SIZE(TCP_MSS+40+PBUF_LINK_HLEN)

PBUF_POOL_BUFSIZE: the size of each pbuf in the pbuf pool. The default is designed to accomodate single full size TCP frame in one pbuf, including TCP_MSS, IP header, and link header.

Definition at line 1104 of file opt.h.

5.72.2.151 #define PBUF_POOL_SIZE 16

PBUF_POOL_SIZE: the number of buffers in the pbuf pool.

Definition at line 414 of file opt.h.

5.72.2.152 #define PPP_DEBUG LWIP_DBG_OFF

PPP_DEBUG: Enable debugging for PPP.

Definition at line 2088 of file opt.h.

5.72.2.153 #define PPP_SUPPORT 0

PPP_SUPPORT==1: Enable PPP.

Definition at line 1627 of file opt.h.

5.72.2.154 #define PPP_THREAD_NAME "ppplnputThread"

PPP_THREAD_NAME: The name assigned to the ppplnputThread.

Definition at line 1300 of file opt.h.

5.72.2.155 #define PPP_THREAD_PRIO 1

PPP_THREAD_PRIO: The priority assigned to the ppplnputThread. The priority value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1318 of file opt.h.

5.72.2.156 #define PPP_THREAD_STACKSIZE 0

PPP_THREAD_STACKSIZE: The stack size used by the pppInputThread. The stack size value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1309 of file opt.h.

5.72.2.157 #define PPPOE_SUPPORT 0

PPPOE_SUPPORT==1: Enable PPP Over Ethernet

Definition at line 1634 of file opt.h.

5.72.2.158 #define PPPOS_SUPPORT PPP_SUPPORT

PPPOS_SUPPORT==1: Enable PPP Over Serial

Definition at line 1641 of file opt.h.

5.72.2.159 #define RAW_DEBUG LWIP_DBG_OFF

RAW_DEBUG: Enable debugging in [raw.c](#).

Definition at line 1975 of file opt.h.

5.72.2.160 #define RAW_TTL (IP_DEFAULT_TTL)

LWIP_RAW==1: Enable application layer to hook into the IP layer itself.

Definition at line 670 of file opt.h.

5.72.2.161 #define RECV_BUFSIZE_DEFAULT INT_MAX

If LWIP_SO_RCVBUF is used, this is the default value for recv_bufsize.

Definition at line 1482 of file opt.h.

5.72.2.162 #define SLIP_DEBUG LWIP_DBG_OFF

SLIP_DEBUG: Enable debugging in [slipif.c](#).

Definition at line 2095 of file opt.h.

5.72.2.163 #define SLIPIF_THREAD_NAME "slipif_loop"

SLIPIF_THREAD_NAME: The name assigned to the slipif_loop thread.

Definition at line 1275 of file opt.h.

5.72.2.164 #define SLIPIF_THREAD_PRIO 1

SLIPIF_THREAD_PRIO: The priority assigned to the slipif_loop thread. The priority value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1293 of file opt.h.

5.72.2.165 #define SLIPIF_THREAD_STACKSIZE 0

SLIP_THREAD_STACKSIZE: The stack size used by the slipif_loop thread. The stack size value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1284 of file opt.h.

5.72.2.166 #define SMEMCPY(dst, src, len) memcpy(dst,src,len)

SMEMCPY: override this with care! Some compilers (e.g. gcc) can inline a call to memcpy() if the length is known at compile time and is small.

Definition at line 92 of file opt.h.

5.72.2.167 #define SNMP_CONCURRENT_REQUESTS 1

SNMP_CONCURRENT_REQUESTS: Number of concurrent requests the module will allow. At least one request buffer is required. Does not have to be changed unless external MIBs answer request asynchronously

Definition at line 742 of file opt.h.

5.72.2.168 #define SNMP_MAX_OCTET_STRING_LEN 127

The maximum length of strings used. This affects the size of MEMP_SNMP_VALUE elements.

Definition at line 776 of file opt.h.

5.72.2.169 #define SNMP_MAX_TREE_DEPTH 15

The maximum depth of the SNMP tree. With private MIBs enabled, this depends on your MIB! This affects the size of MEMP_SNMP_VALUE elements.

Definition at line 785 of file opt.h.

5.72.2.170 #define SNMP_MAX_VALUE_SIZE LWIP_MAX((SNMP_MAX_OCTET_STRING_LEN)+1, sizeof(s32_t)*(SNMP_MAX_TREE_DEPTH))

The size of the MEMP_SNMP_VALUE elements, normally calculated from SNMP_MAX_OCTET_STRING_LEN and SNMP_MAX_TREE_DEPTH.

Definition at line 793 of file opt.h.

5.72.2.171 #define SNMP_MIB_DEBUG LWIP_DBG_OFF

SNMP_MIB_DEBUG: Enable debugging for SNMP MIBs.

Definition at line 2123 of file opt.h.

5.72.2.172 #define SNMP_MSG_DEBUG LWIP_DBG_OFF

SNMP_MSG_DEBUG: Enable debugging for SNMP messages.

Definition at line 2116 of file opt.h.

5.72.2.173 #define SNMP_PRIVATE_MIB 0

SNMP_PRIVATE_MIB: When using a private MIB, you have to create a file 'private_mib.h' that contains a 'struct mib_array_node mib_private' which contains your MIB.

Definition at line 759 of file opt.h.

5.72.2.174 #define SNMP_SAFE_REQUESTS 1

Only allow SNMP write actions that are 'safe' (e.g. disabling netifs is not a safe action and disabled when `SNMP_SAFE_REQUESTS = 1`). Unsafe requests are disabled by default!

Definition at line 768 of file opt.h.

5.72.2.175 #define SNMP_TRAP_DESTINATIONS 1

SNMP_TRAP_DESTINATIONS: Number of trap destinations. At least one trap destination is required

Definition at line 750 of file opt.h.

5.72.2.176 #define SO_REUSE 0

SO_REUSE==1: Enable SO_REUSEADDR option.

Definition at line 1489 of file opt.h.

5.72.2.177 #define SO_REUSE_RXTOALL 0

SO_REUSE_RXTOALL==1: Pass a copy of incoming broadcast/multicast packets to all local matches if SO_REUSEADDR is turned on. WARNING: Adds a memcpy for every packet if passing to more than one pcb!

Definition at line 1498 of file opt.h.

5.72.2.178 #define SOCKETS_DEBUG LWIP_DBG_OFF

SOCKETS_DEBUG: Enable debugging in [sockets.c](#).

Definition at line 1933 of file opt.h.

5.72.2.179 #define SYS_DEBUG LWIP_DBG_OFF

SYS_DEBUG: Enable debugging in [sys.c](#).

Definition at line 1996 of file opt.h.

5.72.2.180 #define SYS_LIGHTWEIGHT_PROT 0

SYS_LIGHTWEIGHT_PROT==1: if you want inter-task protection for certain critical regions during buffer allocation, deallocation and memory allocation and deallocation.

Definition at line 60 of file opt.h.

5.72.2.181 #define SYS_STATS (NO_SYS == 0)

SYS_STATS==1: Enable system stats (sem and mbox counts, etc).

Definition at line 1599 of file opt.h.

5.72.2.182 #define TCP_CALCULATE_EFF_SEND_MSS 1

TCP_CALCULATE_EFF_SEND_MSS: "The maximum size of a segment that TCP really sends, the 'effective send MSS,' MUST be the smaller of the send MSS (which reflects the available reassembly buffer size at the remote host) and the largest size permitted by the IP layer" (RFC 1122) Setting this to 1 enables code that checks TCP_MSS against the MTU of the netif used for a connection and limits the MSS if it would be too big otherwise.

Definition at line 968 of file opt.h.

5.72.2.183 #define TCP_CWND_DEBUG LWIP_DBG_OFF

TCP_CWND_DEBUG: Enable debugging for TCP congestion window.

Definition at line 2039 of file opt.h.

5.72.2.184 #define TCP_DEBUG LWIP_DBG_OFF

TCP_DEBUG: Enable debugging for TCP.

Definition at line 2010 of file opt.h.

5.72.2.185 #define TCP_DEFAULT_LISTEN_BACKLOG 0xff

The maximum allowed backlog for TCP listen netconns. This backlog is used unless another is explicitly specified. 0xff is the maximum (u8_t).

Definition at line 1035 of file opt.h.

5.72.2.186 #define TCP_FR_DEBUG LWIP_DBG_OFF

TCP_FR_DEBUG: Enable debugging in [tcp_in.c](#) for fast retransmit.

Definition at line 2024 of file opt.h.

5.72.2.187 #define TCP_INPUT_DEBUG LWIP_DBG_OFF

TCP_INPUT_DEBUG: Enable debugging in [tcp_in.c](#) for incoming debug.

Definition at line 2017 of file opt.h.

5.72.2.188 #define TCP_LISTEN_BACKLOG 0

TCP_LISTEN_BACKLOG: Enable the backlog option for tcp listen pcb.

Definition at line 1026 of file opt.h.

5.72.2.189 #define TCP_MAXRTX 12

TCP_MAXRTX: Maximum number of retransmissions of data segments.

Definition at line 930 of file opt.h.

5.72.2.190 #define TCP_MSS 536

TCP_MSS: TCP Maximum segment size. (default is 536, a conservative default, you might want to increase this.) For the receive side, this MSS is advertised to the remote side when opening a connection. For the transmit size, this MSS sets an upper limit on the MSS advertised by the remote host.

Definition at line 956 of file opt.h.

5.72.2.191 `#define TCP_OOSEQ_MAX_BYTES 0`

`TCP_OOSEQ_MAX_BYTES`: The maximum number of bytes queued on ooseq per pcb. Default is 0 (no limit). Only valid for `TCP_QUEUE_OOSEQ==0`.

Definition at line 1011 of file opt.h.

5.72.2.192 `#define TCP_OOSEQ_MAX_PBUFS 0`

`TCP_OOSEQ_MAX_PBUFS`: The maximum number of pbufs queued on ooseq per pcb. Default is 0 (no limit). Only valid for `TCP_QUEUE_OOSEQ==0`.

Definition at line 1019 of file opt.h.

5.72.2.193 `#define TCP_OUTPUT_DEBUG LWIP_DBG_OFF`

`TCP_OUTPUT_DEBUG`: Enable debugging in [tcp_out.c](#) output functions.

Definition at line 2053 of file opt.h.

5.72.2.194 `#define TCP_OVERSIZE TCP_MSS`

`TCP_OVERSIZE`: The maximum number of bytes that `tcp_write` may allocate ahead of time in an attempt to create shorter pbuf chains for transmission. The meaningful range is 0 to `TCP_MSS`. Some suggested values are:

0: Disable oversized allocation. Each `tcp_write()` allocates a new pbuf (old behaviour). 1: Allocate size-aligned pbufs with minimal excess. Use this if your scatter-gather DMA requires aligned fragments. 128: Limit the pbuf/memory overhead to 20%. `TCP_MSS`: Try to create unfragmented TCP packets. `TCP_MSS/4`: Try to create 4 fragments or less per TCP packet.

Definition at line 1053 of file opt.h.

5.72.2.195 `#define TCP_QLEN_DEBUG LWIP_DBG_OFF`

`TCP_QLEN_DEBUG`: Enable debugging for TCP queue lengths.

Definition at line 2067 of file opt.h.

5.72.2.196 `#define TCP_QUEUE_OOSEQ (LWIP_TCP)`

`TCP_QUEUE_OOSEQ==1`: TCP will queue segments that arrive out of order. Define to 0 if your device is low on memory.

Definition at line 945 of file opt.h.

5.72.2.197 `#define TCP_RST_DEBUG LWIP_DBG_OFF`

`TCP_RST_DEBUG`: Enable debugging for TCP with the RST message.

Definition at line 2060 of file opt.h.

5.72.2.198 `#define TCP_RTO_DEBUG LWIP_DBG_OFF`

`TCP_RTO_DEBUG`: Enable debugging in TCP for retransmit timeout.

Definition at line 2032 of file opt.h.

5.72.2.199 `#define TCP_SND_BUF (2 * TCP_MSS)`

`TCP_SND_BUF`: TCP sender buffer space (bytes). To achieve good performance, this should be at least $2 * TCP_MSS$.

Definition at line 977 of file opt.h.

5.72.2.200 `#define TCP_SND_QUEUELEN ((4 * (TCP_SND_BUF) + (TCP_MSS - 1))/(TCP_MSS))`

`TCP_SND_QUEUELEN`: TCP sender buffer space (pbufs). This must be at least as much as $(2 * TCP_SND_BUF / TCP_MSS)$ for things to work.

Definition at line 985 of file opt.h.

5.72.2.201 `#define TCP_SNDLOWAT LWIP_MIN(LWIP_MAX(((TCP_SND_BUF)/2), (2 * TCP_MSS) + 1), (TCP_SND_BUF) - 1)`

`TCP_SNDLOWAT`: TCP writable space (bytes). This must be less than `TCP_SND_BUF`. It is the amount of space which must be available in the TCP `snd_buf` for select to return writable (combined with `TCP_SNDQUEUELOWAT`).

Definition at line 994 of file opt.h.

5.72.2.202 `#define TCP_SNDQUEUELOWAT LWIP_MAX(((TCP_SND_QUEUELEN)/2), 5)`

`TCP_SNDQUEUELOWAT`: TCP writable bufs (pbuf count). This must be less than `TCP_SND_QUEUELEN`. If the number of pbufs queued on a pcb drops below this number, select returns writable (combined with `TCP_SNDLOWAT`).

Definition at line 1003 of file opt.h.

5.72.2.203 `#define TCP_STATS (LWIP_TCP)`

`TCP_STATS==1`: Enable TCP stats. Default is on if TCP enabled, otherwise off.

Definition at line 1578 of file opt.h.

5.72.2.204 `#define TCP_SYNMAXRTX 6`

`TCP_SYNMAXRTX`: Maximum number of retransmissions of SYN segments.

Definition at line 937 of file opt.h.

5.72.2.205 `#define TCP_TTL (IP_DEFAULT_TTL)`

`TCP_TTL`: Default Time-To-Live value.

Definition at line 915 of file opt.h.

5.72.2.206 `#define TCP_WND (4 * TCP_MSS)`

`TCP_WND`: The size of a TCP window. This must be at least $(2 * TCP_MSS)$ for things to work well

Definition at line 923 of file opt.h.

5.72.2.207 #define TCP_WND_DEBUG LWIP_DBG_OFF

TCP_WND_DEBUG: Enable debugging in [tcp_in.c](#) for window updating.

Definition at line 2046 of file opt.h.

5.72.2.208 #define TCP_WND_UPDATE_THRESHOLD (TCP_WND / 4)

TCP_WND_UPDATE_THRESHOLD: difference in window to trigger an explicit window update

Definition at line 1068 of file opt.h.

5.72.2.209 #define TCPIP_DEBUG LWIP_DBG_OFF

TCPIP_DEBUG: Enable debugging in [tcpip.c](#).

Definition at line 2081 of file opt.h.

5.72.2.210 #define TCPIP_MBOX_SIZE 0

TCPIP_MBOX_SIZE: The mailbox size for the tcpip thread messages. The queue size value itself is platform-dependent, but is passed to [sys_mbox_new\(\)](#) when `tcpip_init` is called.

Definition at line 1268 of file opt.h.

5.72.2.211 #define TCPIP_THREAD_NAME "tcpip_thread"

TCPIP_THREAD_NAME: The name assigned to the main tcpip thread.

Definition at line 1241 of file opt.h.

5.72.2.212 #define TCPIP_THREAD_PRIO 1

TCPIP_THREAD_PRIO: The priority assigned to the main tcpip thread. The priority value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1259 of file opt.h.

5.72.2.213 #define TCPIP_THREAD_STACKSIZE 0

TCPIP_THREAD_STACKSIZE: The stack size used by the main tcpip thread. The stack size value itself is platform-dependent, but is passed to [sys_thread_new\(\)](#) when the thread is created.

Definition at line 1250 of file opt.h.

5.72.2.214 #define TIMERS_DEBUG LWIP_DBG_OFF

TIMERS_DEBUG: Enable debugging in [timers.c](#).

Definition at line 2003 of file opt.h.

5.72.2.215 #define UDP_DEBUG LWIP_DBG_OFF

UDP_DEBUG: Enable debugging in UDP.

Definition at line 2074 of file opt.h.

5.72.2.216 #define UDP_STATS (LWIP_UDP)

UDP_STATS==1: Enable UDP stats. Default is on if UDP enabled, otherwise off.

Definition at line 1570 of file opt.h.

5.72.2.217 #define UDP_TTL (IP_DEFAULT_TTL)

UDP_TTL: Default Time-To-Live value.

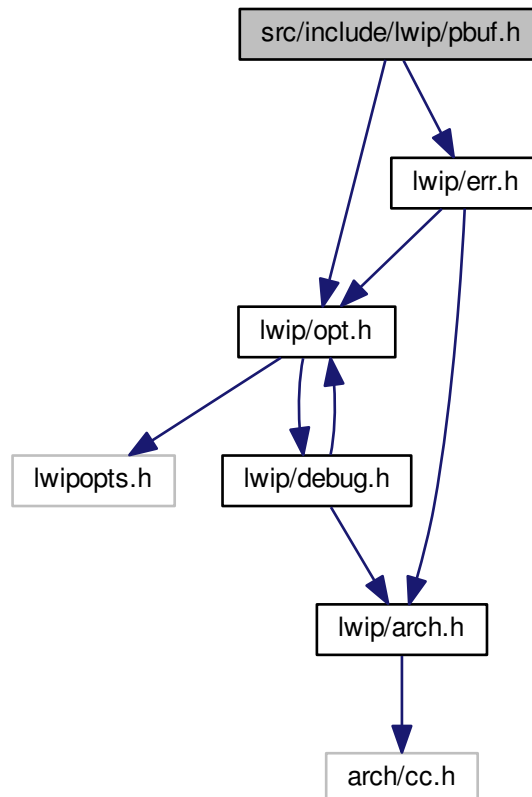
Definition at line 889 of file opt.h.

5.73 src/include/lwip/pbuf.h File Reference

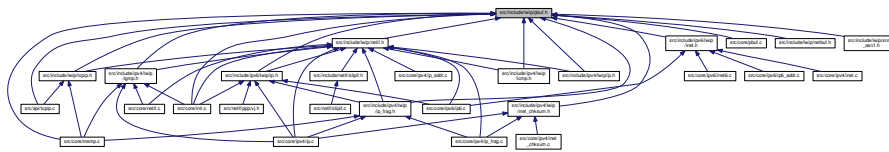
```
#include "lwip/opt.h"
```

```
#include "lwip/err.h"
```

Include dependency graph for pbuf.h:



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



Data Structures

- struct [pbuf](#)

Macros

- `#define LWIP_SUPPORT_CUSTOM_PBUF (IP_FRAG && !IP_FRAG_USES_STATIC_BUF && !LWIP_NETIF_TX_SINGLE_PBUF)`
- `#define PBUF_TRANSPORT_HLEN 20`
- `#define PBUF_IP_HLEN 20`
- `#define PBUF_FLAG_PUSH 0x01U`
- `#define PBUF_FLAG_IS_CUSTOM 0x02U`
- `#define PBUF_FLAG_MCASTLOOP 0x04U`
- `#define PBUF_FLAG_LLBCAST 0x08U`
- `#define PBUF_FLAG_LLMCAST 0x10U`
- `#define PBUF_FLAG_TCP_FIN 0x20U`
- `#define pbuf_init()`

Enumerations

- enum [pbuf_layer](#) { [PBUF_TRANSPORT](#), [PBUF_IP](#), [PBUF_LINK](#), [PBUF_RAW](#) }
- enum [pbuf_type](#) { [PBUF_RAM](#), [PBUF_ROM](#), [PBUF_REF](#), [PBUF_POOL](#) }

Functions

- struct [pbuf](#) * [pbuf_alloc](#) ([pbuf_layer](#) l, u16_t length, [pbuf_type](#) type)
- void [pbuf_realloc](#) (struct [pbuf](#) *p, u16_t size)
- u8_t [pbuf_header](#) (struct [pbuf](#) *p, s16_t header_size)
- void [pbuf_ref](#) (struct [pbuf](#) *p)
- u8_t [pbuf_free](#) (struct [pbuf](#) *p)
- u8_t [pbuf_clen](#) (struct [pbuf](#) *p)
- void [pbuf_cat](#) (struct [pbuf](#) *head, struct [pbuf](#) *tail)
- void [pbuf_chain](#) (struct [pbuf](#) *head, struct [pbuf](#) *tail)
- struct [pbuf](#) * [pbuf_dechain](#) (struct [pbuf](#) *p)
- [err_t](#) [pbuf_copy](#) (struct [pbuf](#) *p_to, struct [pbuf](#) *p_from)
- u16_t [pbuf_copy_partial](#) (struct [pbuf](#) *p, void *dataptr, u16_t len, u16_t offset)
- [err_t](#) [pbuf_take](#) (struct [pbuf](#) *buf, const void *dataptr, u16_t len)
- struct [pbuf](#) * [pbuf_coalesce](#) (struct [pbuf](#) *p, [pbuf_layer](#) layer)
- u8_t [pbuf_get_at](#) (struct [pbuf](#) *p, u16_t offset)
- u16_t [pbuf_memcmp](#) (struct [pbuf](#) *p, u16_t offset, const void *s2, u16_t n)
- u16_t [pbuf_memfind](#) (struct [pbuf](#) *p, const void *mem, u16_t mem_len, u16_t start_offset)
- u16_t [pbuf_strstr](#) (struct [pbuf](#) *p, const char *substr)

5.73.1 Macro Definition Documentation

5.73.1.1 `#define LWIP_SUPPORT_CUSTOM_PBUF (IP_FRAG && !IP_FRAG_USES_STATIC_BUF && !LWIP_NETIF_TX_SINGLE_PBUF)`

Currently, the pbuf_custom code is only needed for one specific configuration of IP_FRAG
Definition at line 45 of file pbuf.h.

5.73.1.2 `#define PBUF_FLAG_IS_CUSTOM 0x02U`

indicates this is a custom pbuf: pbuf_free and pbuf_header handle such a pbuf differently
Definition at line 69 of file pbuf.h.

5.73.1.3 `#define PBUF_FLAG_LLBCAST 0x08U`

indicates this pbuf was received as link-level broadcast
Definition at line 73 of file pbuf.h.

5.73.1.4 `#define PBUF_FLAG_LLMCAST 0x10U`

indicates this pbuf was received as link-level multicast
Definition at line 75 of file pbuf.h.

5.73.1.5 `#define PBUF_FLAG_MCASTLOOP 0x04U`

indicates this pbuf is UDP multicast to be looped back
Definition at line 71 of file pbuf.h.

5.73.1.6 `#define PBUF_FLAG_PUSH 0x01U`

indicates this packet's data should be immediately passed to the application
Definition at line 66 of file pbuf.h.

5.73.1.7 `#define PBUF_FLAG_TCP_FIN 0x20U`

indicates this pbuf includes a TCP FIN flag
Definition at line 77 of file pbuf.h.

5.73.1.8 `#define pbuf_init()`

Definition at line 144 of file pbuf.h.

5.73.1.9 `#define PBUF_IP_HLEN 20`

Definition at line 48 of file pbuf.h.

5.73.1.10 #define PBUF_TRANSPORT_HLEN 20

Definition at line 47 of file pbuf.h.

5.73.2 Enumeration Type Documentation

5.73.2.1 enum pbuf_layer

Enumerator

PBUF_TRANSPORT

PBUF_IP

PBUF_LINK

PBUF_RAW

Definition at line 50 of file pbuf.h.

5.73.2.2 enum pbuf_type

Enumerator

PBUF_RAM

PBUF_ROM

PBUF_REF

PBUF_POOL

Definition at line 57 of file pbuf.h.

5.73.3 Function Documentation

5.73.3.1 struct pbuf* pbuf_alloc (pbuf_layer layer, u16_t length, pbuf_type type)

Allocates a pbuf of the given type (possibly a chain for PBUF_POOL type).

The actual memory allocated for the pbuf is determined by the layer at which the pbuf is allocated and the requested size (from the size parameter).

Parameters

<i>layer</i>	flag to define header size
<i>length</i>	size of the pbuf's payload
<i>type</i>	this parameter decides how and where the pbuf should be allocated as follows:

- PBUF_RAM: buffer memory for pbuf is allocated as one large chunk. This includes protocol headers as well.
- PBUF_ROM: no buffer memory is allocated for the pbuf, even for protocol headers. Additional headers must be prepended by allocating another pbuf and chain in to the front of the ROM pbuf. It is assumed that the memory used is really similar to ROM in that it is immutable and will not be changed. Memory which is dynamic should generally not be attached to PBUF_ROM pbufs. Use PBUF_REF instead.
- PBUF_REF: no buffer memory is allocated for the pbuf, even for protocol headers. It is assumed that the pbuf is only being used in a single thread. If the pbuf gets queued, then pbuf_take should be called to copy the buffer.
- PBUF_POOL: the pbuf is allocated as a pbuf chain, with pbufs from the pbuf pool that is allocated during [pbuf_init\(\)](#).

Returns

the allocated pbuf. If multiple pbufs were allocated, this is the first pbuf of a pbuf chain.

Definition at line 207 of file pbuf.c.

5.73.3.2 void pbuf_cat (struct pbuf * *h*, struct pbuf * *t*)

Concatenate two pbufs (each may be a pbuf chain) and take over the caller's reference of the tail pbuf.

Note

The caller MAY NOT reference the tail pbuf afterwards. Use [pbuf_chain\(\)](#) for that purpose.

See also

[pbuf_chain\(\)](#)

Definition at line 745 of file pbuf.c.

5.73.3.3 void pbuf_chain (struct pbuf * *h*, struct pbuf * *t*)

Chain two pbufs (or pbuf chains) together.

The caller MUST call pbuf_free(t) once it has stopped using it. Use [pbuf_cat\(\)](#) instead if you no longer use t.

Parameters

<i>h</i>	head pbuf (chain)
<i>t</i>	tail pbuf (chain)

Note

The pbufs MUST belong to the same packet.
MAY NOT be called on a packet queue.

The ->tot_len fields of all pbufs of the head chain are adjusted. The ->next field of the last pbuf of the head chain is adjusted. The ->ref field of the first pbuf of the tail chain is adjusted.

Definition at line 786 of file pbuf.c.

5.73.3.4 u8_t pbuf_clen (struct pbuf * *p*)

Count number of pbufs in a chain

Parameters

<i>p</i>	first pbuf of chain
----------	---------------------

Returns

the number of pbufs in a chain

Definition at line 704 of file pbuf.c.

5.73.3.5 struct pbuf* pbuf_coalesce (struct pbuf * *p*, pbuf_layer *layer*)

Creates a single pbuf out of a queue of pbufs.

Remarks

: Either the source pbuf 'p' is freed by this function or the original pbuf 'p' is returned, therefore the caller has to check the result!

Parameters

<i>p</i>	the source pbuf
<i>layer</i>	pbuf_layer of the new pbuf

Returns

a new, single pbuf (p->next is NULL) or the old pbuf if allocation fails

Definition at line 1010 of file pbuf.c.

5.73.3.6 err_t pbuf_copy (struct pbuf * *p_to*, struct pbuf * *p_from*)

Create PBUF_RAM copies of pbufs.

Used to queue packets on behalf of the lwIP stack, such as ARP based queueing.

Note

You MUST explicitly use `p = pbuf_take(p);`
Only one packet is copied, no packet queue!

Parameters

<i>p_to</i>	pbuf destination of the copy
<i>p_from</i>	pbuf source of the copy

Returns

ERR_OK if pbuf was copied ERR_ARG if one of the pbufs is NULL or p_to is not big enough to hold p_from

Definition at line 852 of file pbuf.c.

5.73.3.7 u16_t pbuf_copy_partial (struct pbuf * *buf*, void * *dataptr*, u16_t *len*, u16_t *offset*)

Copy (part of) the contents of a packet buffer to an application supplied buffer.

Parameters

<i>buf</i>	the pbuf from which to copy data
<i>dataptr</i>	the application supplied buffer
<i>len</i>	length of data to copy (dataptr must be big enough). No more than buf->tot_len will be copied, irrespective of len
<i>offset</i>	offset into the packet buffer from where to begin copying len bytes

Returns

the number of bytes copied, or 0 on failure

Definition at line 918 of file pbuf.c.

5.73.3.8 struct pbuf* pbuf_dechain (struct pbuf * p)

Dechains the first pbuf from its succeeding pbufs in the chain.

Makes p->tot_len field equal to p->len.

Parameters

<i>p</i>	pbuf to dechain
----------	-----------------

Returns

remainder of the pbuf chain, or NULL if it was de-allocated.

Note

May not be called on a packet queue.

Definition at line 803 of file pbuf.c.

5.73.3.9 u8_t pbuf_free (struct pbuf * p)

Dereference a pbuf chain or queue and deallocate any no-longer-used pbufs at the head of this chain or queue.

Decrements the pbuf reference count. If it reaches zero, the pbuf is deallocated.

For a pbuf chain, this is repeated for each pbuf in the chain, up to the first pbuf which has a non-zero reference count after decrementing. So, when all reference counts are one, the whole chain is free'd.

Parameters

<i>p</i>	The pbuf (chain) to be dereferenced.
----------	--------------------------------------

Returns

the number of pbufs that were de-allocated from the head of the chain.

Note

MUST NOT be called on a packet queue (Not verified to work yet).

the reference counter of a pbuf equals the number of pointers that refer to the pbuf (or into the pbuf).

Definition at line 618 of file pbuf.c.

5.73.3.10 u8_t pbuf_get_at (struct pbuf * p, u16_t offset)

Get one byte from the specified position in a pbuf WARNING: returns zero for offset >= p->tot_len

Parameters

<i>p</i>	pbuf to parse
<i>offset</i>	offset into p of the byte to return

Returns

byte at an offset into p OR ZERO IF 'offset' >= p->tot_len

Definition at line 1077 of file pbuf.c.

5.73.3.11 `u8_t pbuf_header (struct pbuf * p, s16_t header_size_increment)`

Adjusts the payload pointer to hide or reveal headers in the payload.

Adjusts the ->payload pointer so that space for a header (dis)appears in the pbuf payload.

The ->payload, ->tot_len and ->len fields are adjusted.

Parameters

<i>p</i>	pbuf to change the header size.
<i>header_size_increment</i>	Number of bytes to increment header size which increases the size of the pbuf. New space is on the front. (Using a negative value decreases the header size.) If hdr_size_inc is 0, this function does nothing and returns successful.

PBUF_ROM and PBUF_REF type buffers cannot have their sizes increased, so the call will fail. A check is made that the increase in header size does not move the payload pointer in front of the start of the buffer.

Returns

non-zero on failure, zero on success.

Definition at line 511 of file pbuf.c.

5.73.3.12 `u16_t pbuf_memcmp (struct pbuf * p, u16_t offset, const void * s2, u16_t n)`

Compare pbuf contents at specified offset with memory s2, both of length n

Parameters

<i>p</i>	pbuf to compare
<i>offset</i>	offset into p at which to start comparing
<i>s2</i>	buffer to compare
<i>n</i>	length of buffer to compare

Returns

zero if equal, nonzero otherwise (0xffff if p is too short, diff+offset+1 otherwise)

Definition at line 1104 of file pbuf.c.

5.73.3.13 `u16_t pbuf_memfind (struct pbuf * p, const void * mem, u16_t mem_len, u16_t start_offset)`

Find occurrence of mem (with length mem_len) in pbuf p, starting at offset start_offset.

Parameters

<i>p</i>	pbuf to search, maximum length is 0xFFFE since 0xFFFF is used as return value 'not found'
<i>mem</i>	search for the contents of this buffer
<i>mem_len</i>	length of 'mem'
<i>start_offset</i>	offset into p at which to start searching

Returns

0xFFFF if substr was not found in p or the index where it was found

Definition at line 1140 of file pbuf.c.

5.73.3.14 `void pbuf_realloc (struct pbuf * p, u16_t new_len)`

Shrink a pbuf chain to a desired length.

Parameters

<i>p</i>	pbuf to shrink.
<i>new_len</i>	desired new length of pbuf chain

Depending on the desired length, the first few pbufs in a chain might be skipped and left unchanged. The new last pbuf in the chain will be resized, and any remaining pbufs will be freed.

Note

If the pbuf is ROM/REF, only the ->tot_len and ->len fields are adjusted.
May not be called on a packet queue.
Despite its name, pbuf_realloc cannot grow the size of a pbuf (chain).

Definition at line 430 of file pbuf.c.

5.73.3.15 void pbuf_ref (struct pbuf * *p*)

Increment the reference count of the pbuf.

Parameters

<i>p</i>	pbuf to increase reference counter of
----------	---------------------------------------

Definition at line 723 of file pbuf.c.

5.73.3.16 u16_t pbuf_strstr (struct pbuf * *p*, const char * *substr*)

Find occurrence of substr with length substr_len in pbuf *p*, start at offset start_offset WARNING: in contrast to strstr(), this one does not stop at the first \0 in the pbuf/source string!

Parameters

<i>p</i>	pbuf to search, maximum length is 0xFFFE since 0xFFFF is used as return value 'not found'
<i>substr</i>	string to search for in <i>p</i> , maximum length is 0xFFFE

Returns

0xFFFF if substr was not found in *p* or the index where it was found

Definition at line 1168 of file pbuf.c.

5.73.3.17 err_t pbuf_take (struct pbuf * *buf*, const void * *dataptr*, u16_t *len*)

Copy application supplied data into a pbuf. This function can only be used to copy the equivalent of buf->tot_len data.

Parameters

<i>buf</i>	pbuf to fill with data
<i>dataptr</i>	application supplied data buffer
<i>len</i>	length of the application supplied data buffer

Returns

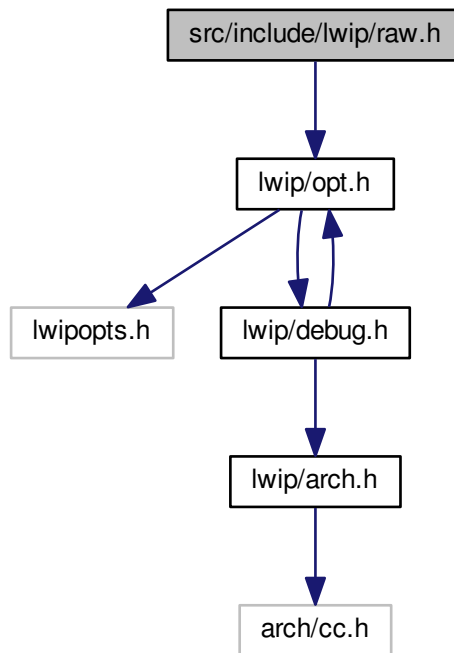
ERR_OK if successful, ERR_MEM if the pbuf is not big enough

Definition at line 966 of file pbuf.c.

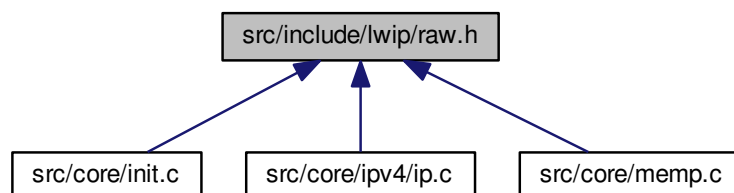
5.74 src/include/lwip/raw.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for raw.h:



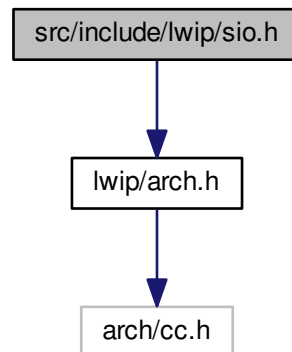
This graph shows which files directly or indirectly include this file:



5.75 src/include/lwip/sio.h File Reference

```
#include "lwip/arch.h"
```

Include dependency graph for sio.h:



Typedefs

- typedef void * [sio_fd_t](#)

Functions

- [sio_fd_t sio_open](#) (u8_t devnum)
- void [sio_send](#) (u8_t c, [sio_fd_t](#) fd)
- u8_t [sio_recv](#) ([sio_fd_t](#) fd)
- u32_t [sio_read](#) ([sio_fd_t](#) fd, u8_t *data, u32_t len)
- u32_t [sio_tryread](#) ([sio_fd_t](#) fd, u8_t *data, u32_t len)
- u32_t [sio_write](#) ([sio_fd_t](#) fd, u8_t *data, u32_t len)
- void [sio_read_abort](#) ([sio_fd_t](#) fd)

5.75.1 Typedef Documentation

5.75.1.1 typedef void* [sio_fd_t](#)

Definition at line 47 of file sio.h.

5.75.2 Function Documentation

5.75.2.1 [sio_fd_t sio_open](#) (u8_t *devnum*)

Opens a serial device for communication.

Parameters

<i>devnum</i>	device number
---------------	---------------

Returns

handle to serial device if successful, NULL otherwise

5.75.2.2 `u32_t sio_read (sio_fd_t fd, u8_t * data, u32_t len)`

Reads from the serial device.

Parameters

<i>fd</i>	serial device handle
<i>data</i>	pointer to data buffer for receiving
<i>len</i>	maximum length (in bytes) of data to receive

Returns

number of bytes actually received - may be 0 if aborted by `sio_read_abort`

Note

This function will block until data can be received. The blocking can be cancelled by calling [sio_read_abort\(\)](#).

5.75.2.3 void sio_read_abort (sio_fd_t fd)

Aborts a blocking [sio_read\(\)](#) call.

Parameters

<i>fd</i>	serial device handle
-----------	----------------------

5.75.2.4 u8_t sio_recv (sio_fd_t fd)

Receives a single character from the serial device.

Parameters

<i>fd</i>	serial device handle
-----------	----------------------

Note

This function will block until a character is received.

5.75.2.5 void sio_send (u8_t c, sio_fd_t fd)

Sends a single character to the serial device.

Parameters

<i>c</i>	character to send
<i>fd</i>	serial device handle

Note

This function will block until the character can be sent.

5.75.2.6 u32_t sio_tryread (sio_fd_t fd, u8_t * data, u32_t len)

Tries to read from the serial device. Same as `sio_read` but returns immediately if no data is available and never blocks.

Parameters

<i>fd</i>	serial device handle
<i>data</i>	pointer to data buffer for receiving
<i>len</i>	maximum length (in bytes) of data to receive

Returns

number of bytes actually received

5.75.2.7 u32_t sio_write (sio_fd_t fd, u8_t * data, u32_t len)

Writes to the serial device.

Parameters

<i>fd</i>	serial device handle
<i>data</i>	pointer to data to send
<i>len</i>	length (in bytes) of data to send

Returns

number of bytes actually sent

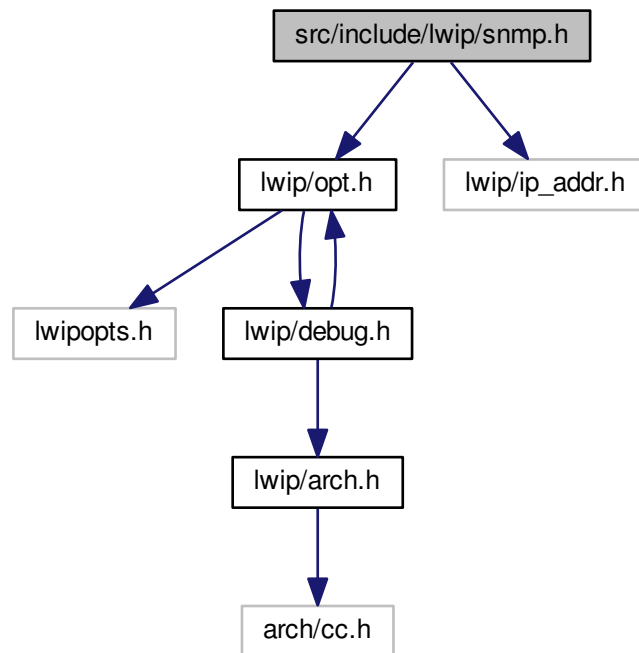
Note

This function will block until all data can be sent.

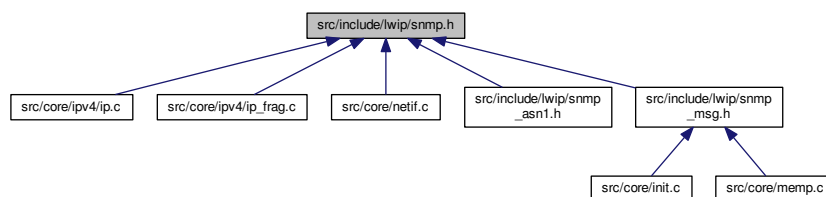
5.76 src/include/lwip/snmp.h File Reference

```
#include "lwip/opt.h"
#include "lwip/ip_addr.h"
```


Include dependency graph for snmp.h:



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



Macros

- `#define snmp_set_sysdesr(str, len)`
- `#define snmp_set_sysobjid(oid) ;`
- `#define snmp_get_sysobjid_ptr(oid)`
- `#define snmp_inc_sysuptime()`
- `#define snmp_add_sysuptime(value)`
- `#define snmp_get_sysuptime(value)`
- `#define snmp_set_syscontact(ocstr, ocstrlen) ;`
- `#define snmp_set_sysname(ocstr, ocstrlen) ;`
- `#define snmp_set_syslocation(ocstr, ocstrlen) ;`
- `#define snmp_add_ifinoctets(ni, value)`

- #define [snmp_inc_ifinucastpkts](#)(ni)
- #define [snmp_inc_ifinnucastpkts](#)(ni)
- #define [snmp_inc_ifindiscards](#)(ni)
- #define [snmp_add_ifoutoctets](#)(ni, value)
- #define [snmp_inc_ifoutucastpkts](#)(ni)
- #define [snmp_inc_ifoutnucastpkts](#)(ni)
- #define [snmp_inc_ifoutdiscards](#)(ni)
- #define [snmp_inc_iflist](#)()
- #define [snmp_dec_iflist](#)()
- #define [snmp_insert_arpidx_tree](#)(ni, ip)
- #define [snmp_delete_arpidx_tree](#)(ni, ip)
- #define [snmp_inc_ipinreceives](#)()
- #define [snmp_inc_ipinhdrrerrors](#)()
- #define [snmp_inc_ipinaddrerrors](#)()
- #define [snmp_inc_ipforwdatagrams](#)()
- #define [snmp_inc_ipinunknownprotos](#)()
- #define [snmp_inc_ipindiscards](#)()
- #define [snmp_inc_ipindelivers](#)()
- #define [snmp_inc_ipoutrequests](#)()
- #define [snmp_inc_ipoutdiscards](#)()
- #define [snmp_inc_ipoutnoroutes](#)()
- #define [snmp_inc_ippreasmreqds](#)()
- #define [snmp_inc_ippreasmoks](#)()
- #define [snmp_inc_ippreasmfails](#)()
- #define [snmp_inc_ipfragoks](#)()
- #define [snmp_inc_ipfragfails](#)()
- #define [snmp_inc_ipfragcreates](#)()
- #define [snmp_inc_iproutingdiscards](#)()
- #define [snmp_insert_ipaddridx_tree](#)(ni)
- #define [snmp_delete_ipaddridx_tree](#)(ni)
- #define [snmp_insert_iprteidx_tree](#)(dflt, ni)
- #define [snmp_delete_iprteidx_tree](#)(dflt, ni)
- #define [snmp_inc_icmpinmsgs](#)()
- #define [snmp_inc_icmpinerrors](#)()
- #define [snmp_inc_icmpindestunreachs](#)()
- #define [snmp_inc_icmpintimeexcds](#)()
- #define [snmp_inc_icmpinparmprobs](#)()
- #define [snmp_inc_icmpinsrcquenchs](#)()
- #define [snmp_inc_icmpinredirects](#)()
- #define [snmp_inc_icmpinechos](#)()
- #define [snmp_inc_icmpinechoreps](#)()
- #define [snmp_inc_icmpintimestamps](#)()
- #define [snmp_inc_icmpintimestamppreps](#)()
- #define [snmp_inc_icmpinaddrmask](#)s()
- #define [snmp_inc_icmpinaddrmaskreps](#)()
- #define [snmp_inc_icmpoutmsgs](#)()
- #define [snmp_inc_icmpouterrors](#)()
- #define [snmp_inc_icmpoutdestunreachs](#)()
- #define [snmp_inc_icmpouttimeexcds](#)()
- #define [snmp_inc_icmpoutparmprobs](#)()
- #define [snmp_inc_icmpoutsrcquenchs](#)()
- #define [snmp_inc_icmpoutredirects](#)()
- #define [snmp_inc_icmpoutechos](#)()
- #define [snmp_inc_icmpoutechoreps](#)()
- #define [snmp_inc_icmpouttimestamps](#)()

- `#define snmp_inc_icmpouttimestampreps()`
- `#define snmp_inc_icmpoutaddrmaskreps()`
- `#define snmp_inc_icmpoutaddrmaskreps()`
- `#define snmp_inc_tcpactiveopens()`
- `#define snmp_inc_tcppassiveopens()`
- `#define snmp_inc_tcpattemptfails()`
- `#define snmp_inc_tcestabresets()`
- `#define snmp_inc_tcpinsegs()`
- `#define snmp_inc_tcpoutsegs()`
- `#define snmp_inc_tcpretranssegs()`
- `#define snmp_inc_tcpinerrs()`
- `#define snmp_inc_tcpoutrsts()`
- `#define snmp_inc_udpin datagrams()`
- `#define snmp_inc_udpnoports()`
- `#define snmp_inc_udpinerrors()`
- `#define snmp_inc_udpout datagrams()`
- `#define snmp_insert_udpidx_tree(pcb)`
- `#define snmp_delete_udpidx_tree(pcb)`
- `#define snmp_inc_snmpinpkts()`
- `#define snmp_inc_snmpoutpkts()`
- `#define snmp_inc_snmpinbadversions()`
- `#define snmp_inc_snmpinbadcommunitynames()`
- `#define snmp_inc_snmpinbadcommunityuses()`
- `#define snmp_inc_snmpinasnparseerrs()`
- `#define snmp_inc_snmpintoobigs()`
- `#define snmp_inc_snmpinnosuchnames()`
- `#define snmp_inc_snmpinbadvalues()`
- `#define snmp_inc_snmpinreadonlys()`
- `#define snmp_inc_snmpingenerrs()`
- `#define snmp_add_snmpintotalreqvars(value)`
- `#define snmp_add_snmpintotalsetvars(value)`
- `#define snmp_inc_snmpingetrequests()`
- `#define snmp_inc_snmpingetnexts()`
- `#define snmp_inc_snmpinsetrequests()`
- `#define snmp_inc_snmpingetresponses()`
- `#define snmp_inc_snmpintraps()`
- `#define snmp_inc_snmpouttoobigs()`
- `#define snmp_inc_snmpoutnosuchnames()`
- `#define snmp_inc_snmpoutbadvalues()`
- `#define snmp_inc_snmpoutgenerrs()`
- `#define snmp_inc_snmpoutgetrequests()`
- `#define snmp_inc_snmpoutgetnexts()`
- `#define snmp_inc_snmpoutsetrequests()`
- `#define snmp_inc_snmpoutgetresponses()`
- `#define snmp_inc_snmpouttraps()`
- `#define snmp_get_snmpgrpid_ptr(oid)`
- `#define snmp_set_snmpenableauthentraps(value)`
- `#define snmp_get_snmpenableauthentraps(value)`

Enumerations

- enum `snmp_ifType` {
`snmp_ifType_other` =1, `snmp_ifType_regular1822`, `snmp_ifType_hdhd1822`, `snmp_ifType_ddn_x25`,
`snmp_ifType_rfc877_x25`, `snmp_ifType_ethernet_csmacd`, `snmp_ifType_iso88023_csmacd`, `snmp_ifType_iso88024_tokenBus`,
`snmp_ifType_iso88025_tokenRing`, `snmp_ifType_iso88026_man`, `snmp_ifType_starLan`, `snmp_ifType_proteon_10Mbit`,
`snmp_ifType_proteon_80Mbit`, `snmp_ifType_hyperchannel`, `snmp_ifType_fddi`, `snmp_ifType_lapb`,
`snmp_ifType_sdlc`, `snmp_ifType_ds1`, `snmp_ifType_e1`, `snmp_ifType_basicISDN`,
`snmp_ifType_primaryISDN`, `snmp_ifType_propPointToPointSerial`, `snmp_ifType_ppp`, `snmp_ifType_softwareLoopback`,
`snmp_ifType_eon`, `snmp_ifType_ethernet_3Mbit`, `snmp_ifType_nsip`, `snmp_ifType_slip`,
`snmp_ifType_ultra`, `snmp_ifType_ds3`, `snmp_ifType_sip`, `snmp_ifType_frame_relay` }

5.76.1 Macro Definition Documentation

5.76.1.1 `#define snmp_add_ifinOctets(ni, value)`

Definition at line 245 of file `snmp.h`.

5.76.1.2 `#define snmp_add_ifoutOctets(ni, value)`

Definition at line 249 of file `snmp.h`.

5.76.1.3 `#define snmp_add_snmpinttotalreqvars(value)`

Definition at line 341 of file `snmp.h`.

5.76.1.4 `#define snmp_add_snmpinttotalsetvars(value)`

Definition at line 342 of file `snmp.h`.

5.76.1.5 `#define snmp_add_sysuptime(value)`

Definition at line 238 of file `snmp.h`.

5.76.1.6 `#define snmp_dec_iflist()`

Definition at line 254 of file `snmp.h`.

5.76.1.7 `#define snmp_delete_arpidx_tree(ni, ip)`

Definition at line 258 of file `snmp.h`.

5.76.1.8 `#define snmp_delete_ipaddridx_tree(ni)`

Definition at line 279 of file `snmp.h`.

5.76.1.9 `#define snmp_delete_iprtidx_tree(dflt, ni)`

Definition at line 281 of file `snmp.h`.

5.76.1.10 `#define snmp_delete_udpidx_tree(pcb)`

Definition at line 327 of file snmp.h.

5.76.1.11 `#define snmp_get_snmpenableauthentrap(value)`

Definition at line 359 of file snmp.h.

5.76.1.12 `#define snmp_get_snmpgrpid_ptr(oid)`

Definition at line 357 of file snmp.h.

5.76.1.13 `#define snmp_get_sysobjid_ptr(oid)`

Definition at line 236 of file snmp.h.

5.76.1.14 `#define snmp_get_sysuptime(value)`

Definition at line 239 of file snmp.h.

5.76.1.15 `#define snmp_inc_icmpinaddrmaskreps()`

Definition at line 296 of file snmp.h.

5.76.1.16 `#define snmp_inc_icmpinaddrmask()`

Definition at line 295 of file snmp.h.

5.76.1.17 `#define snmp_inc_icmpindestunreachs()`

Definition at line 286 of file snmp.h.

5.76.1.18 `#define snmp_inc_icmpinechoreps()`

Definition at line 292 of file snmp.h.

5.76.1.19 `#define snmp_inc_icmpinechos()`

Definition at line 291 of file snmp.h.

5.76.1.20 `#define snmp_inc_icmpinerrors()`

Definition at line 285 of file snmp.h.

5.76.1.21 `#define snmp_inc_icmpinmsgs()`

Definition at line 284 of file snmp.h.

5.76.1.22 `#define snmp_inc_icmpinparmprobs()`

Definition at line 288 of file snmp.h.

5.76.1.23 `#define snmp_inc_icmpinredirects()`

Definition at line 290 of file snmp.h.

5.76.1.24 `#define snmp_inc_icmpinsrcquenches()`

Definition at line 289 of file snmp.h.

5.76.1.25 `#define snmp_inc_icmpintimeexcds()`

Definition at line 287 of file snmp.h.

5.76.1.26 `#define snmp_inc_icmpintimestampreps()`

Definition at line 294 of file snmp.h.

5.76.1.27 `#define snmp_inc_icmpintimestamps()`

Definition at line 293 of file snmp.h.

5.76.1.28 `#define snmp_inc_icmpoutaddrmaskreps()`

Definition at line 309 of file snmp.h.

5.76.1.29 `#define snmp_inc_icmpoutaddrmasks()`

Definition at line 308 of file snmp.h.

5.76.1.30 `#define snmp_inc_icmpoutdestunreachs()`

Definition at line 299 of file snmp.h.

5.76.1.31 `#define snmp_inc_icmpoutechoreps()`

Definition at line 305 of file snmp.h.

5.76.1.32 `#define snmp_inc_icmpoutechos()`

Definition at line 304 of file snmp.h.

5.76.1.33 `#define snmp_inc_icmpouterrors()`

Definition at line 298 of file snmp.h.

5.76.1.34 `#define snmp_inc_icmpoutmsgs()`

Definition at line 297 of file snmp.h.

5.76.1.35 `#define snmp_inc_icmpoutparmpkts()`

Definition at line 301 of file snmp.h.

5.76.1.36 `#define snmp_inc_icmpoutredirects()`

Definition at line 303 of file snmp.h.

5.76.1.37 `#define snmp_inc_icmpoutsrcquenchs()`

Definition at line 302 of file snmp.h.

5.76.1.38 `#define snmp_inc_icmpouttimeexcds()`

Definition at line 300 of file snmp.h.

5.76.1.39 `#define snmp_inc_icmpouttimestamppreps()`

Definition at line 307 of file snmp.h.

5.76.1.40 `#define snmp_inc_icmpouttimestamps()`

Definition at line 306 of file snmp.h.

5.76.1.41 `#define snmp_inc_ifindiscards(ni)`

Definition at line 248 of file snmp.h.

5.76.1.42 `#define snmp_inc_ifinnucastpkts(ni)`

Definition at line 247 of file snmp.h.

5.76.1.43 `#define snmp_inc_ifinucastpkts(ni)`

Definition at line 246 of file snmp.h.

5.76.1.44 `#define snmp_inc_iflist()`

Definition at line 253 of file snmp.h.

5.76.1.45 `#define snmp_inc_ifoutdiscards(ni)`

Definition at line 252 of file snmp.h.

5.76.1.46 `#define snmp_inc_ifoutnucastpkts(ni)`

Definition at line 251 of file snmp.h.

5.76.1.47 `#define snmp_inc_ifoutucastpkts(ni)`

Definition at line 250 of file snmp.h.

5.76.1.48 `#define snmp_inc_ipforwdatagrams()`

Definition at line 264 of file snmp.h.

5.76.1.49 `#define snmp_inc_ipfragcreates()`

Definition at line 276 of file snmp.h.

5.76.1.50 `#define snmp_inc_ipfragfails()`

Definition at line 275 of file snmp.h.

5.76.1.51 `#define snmp_inc_ipfragoks()`

Definition at line 274 of file snmp.h.

5.76.1.52 `#define snmp_inc_ipinaddrerrors()`

Definition at line 263 of file snmp.h.

5.76.1.53 `#define snmp_inc_ipindelivers()`

Definition at line 267 of file snmp.h.

5.76.1.54 `#define snmp_inc_ipindiscards()`

Definition at line 266 of file snmp.h.

5.76.1.55 `#define snmp_inc_ipinhdrrerrors()`

Definition at line 262 of file snmp.h.

5.76.1.56 `#define snmp_inc_ipinreceives()`

Definition at line 261 of file snmp.h.

5.76.1.57 `#define snmp_inc_ipinunknownprotos()`

Definition at line 265 of file snmp.h.

5.76.1.58 `#define snmp_inc_ipoutdiscards()`

Definition at line 269 of file snmp.h.

5.76.1.59 `#define snmp_inc_ipoutnoroutes()`

Definition at line 270 of file snmp.h.

5.76.1.60 `#define snmp_inc_ipoutrequests()`

Definition at line 268 of file snmp.h.

5.76.1.61 `#define snmp_inc_ipreasmfails()`

Definition at line 273 of file snmp.h.

5.76.1.62 `#define snmp_inc_ipreasmoks()`

Definition at line 272 of file snmp.h.

5.76.1.63 `#define snmp_inc_ipreasmreqds()`

Definition at line 271 of file snmp.h.

5.76.1.64 `#define snmp_inc_iproutingdiscards()`

Definition at line 277 of file snmp.h.

5.76.1.65 `#define snmp_inc_snmpinasnparseerrs()`

Definition at line 335 of file snmp.h.

5.76.1.66 `#define snmp_inc_snmpinbadcommunitynames()`

Definition at line 333 of file snmp.h.

5.76.1.67 `#define snmp_inc_snmpinbadcommunityuses()`

Definition at line 334 of file snmp.h.

5.76.1.68 `#define snmp_inc_snmpinbadvalues()`

Definition at line 338 of file snmp.h.

5.76.1.69 `#define snmp_inc_snmpinbadversions()`

Definition at line 332 of file snmp.h.

5.76.1.70 `#define snmp_inc_snmpingenerrs()`

Definition at line 340 of file snmp.h.

5.76.1.71 `#define snmp_inc_snmpingetnexts()`

Definition at line 344 of file snmp.h.

5.76.1.72 `#define snmp_inc_snmpingetrequests()`

Definition at line 343 of file snmp.h.

5.76.1.73 `#define snmp_inc_snmpingetresponses()`

Definition at line 346 of file snmp.h.

5.76.1.74 `#define snmp_inc_snmpinnosuchnames()`

Definition at line 337 of file snmp.h.

5.76.1.75 `#define snmp_inc_snmpinpkts()`

Definition at line 330 of file snmp.h.

5.76.1.76 `#define snmp_inc_snmpinreadonlys()`

Definition at line 339 of file snmp.h.

5.76.1.77 `#define snmp_inc_snmpinsetrequests()`

Definition at line 345 of file snmp.h.

5.76.1.78 `#define snmp_inc_snmpintoobigs()`

Definition at line 336 of file snmp.h.

5.76.1.79 `#define snmp_inc_snmpintraps()`

Definition at line 347 of file snmp.h.

5.76.1.80 `#define snmp_inc_snmpoutbadvalues()`

Definition at line 350 of file snmp.h.

5.76.1.81 `#define snmp_inc_snmpoutgenerrs()`

Definition at line 351 of file snmp.h.

5.76.1.82 `#define snmp_inc_snmpoutgetnexts()`

Definition at line 353 of file snmp.h.

5.76.1.83 `#define snmp_inc_snmpoutgetrequests()`

Definition at line 352 of file snmp.h.

5.76.1.84 `#define snmp_inc_snmpoutgetresponses()`

Definition at line 355 of file snmp.h.

5.76.1.85 `#define snmp_inc_snmpoutnosuchnames()`

Definition at line 349 of file snmp.h.

5.76.1.86 `#define snmp_inc_snmpoutpkts()`

Definition at line 331 of file snmp.h.

5.76.1.87 `#define snmp_inc_snmpoutsetrequests()`

Definition at line 354 of file snmp.h.

5.76.1.88 `#define snmp_inc_snmpouttoobigs()`

Definition at line 348 of file snmp.h.

5.76.1.89 `#define snmp_inc_snmpouttraps()`

Definition at line 356 of file snmp.h.

5.76.1.90 `#define snmp_inc_sysuptime()`

Definition at line 237 of file snmp.h.

5.76.1.91 `#define snmp_inc_tcpactiveopens()`

Definition at line 311 of file snmp.h.

5.76.1.92 `#define snmp_inc_tcpattemptfails()`

Definition at line 313 of file snmp.h.

5.76.1.93 `#define snmp_inc_tcpstabresets()`

Definition at line 314 of file snmp.h.

5.76.1.94 `#define snmp_inc_tcpinerrs()`

Definition at line 318 of file snmp.h.

5.76.1.95 `#define snmp_inc_tcpinsegs()`

Definition at line 315 of file snmp.h.

5.76.1.96 `#define snmp_inc_tcpouttrsts()`

Definition at line 319 of file snmp.h.

5.76.1.97 `#define snmp_inc_tcpoutsegs()`

Definition at line 316 of file snmp.h.

5.76.1.98 `#define snmp_inc_tcppassiveopens()`

Definition at line 312 of file snmp.h.

5.76.1.99 `#define snmp_inc_tcpretranssegs()`

Definition at line 317 of file snmp.h.

5.76.1.100 `#define snmp_inc_udpindatagrams()`

Definition at line 322 of file snmp.h.

5.76.1.101 `#define snmp_inc_udpinerrors()`

Definition at line 324 of file snmp.h.

5.76.1.102 `#define snmp_inc_udpnoports()`

Definition at line 323 of file snmp.h.

5.76.1.103 `#define snmp_inc_udpoutdatagrams()`

Definition at line 325 of file snmp.h.

5.76.1.104 `#define snmp_insert_arpidx_tree(ni, ip)`

Definition at line 257 of file snmp.h.

5.76.1.105 `#define snmp_insert_ipaddridx_tree(ni)`

Definition at line 278 of file snmp.h.

5.76.1.106 `#define snmp_insert_iprtidx_tree(dflt, ni)`

Definition at line 280 of file snmp.h.

5.76.1.107 `#define snmp_insert_udpidx_tree(pcb)`

Definition at line 326 of file snmp.h.

5.76.1.108 `#define snmp_set_snmpenableauthentraps(value)`

Definition at line 358 of file snmp.h.

5.76.1.109 `#define snmp_set_syscontact(ocstr, ocstrlen);`

Definition at line 240 of file snmp.h.

5.76.1.110 `#define snmp_set_sysdesr(str, len)`

Definition at line 234 of file snmp.h.

5.76.1.111 `#define snmp_set_syslocation(ocstr, ocstrlen);`

Definition at line 242 of file snmp.h.

5.76.1.112 `#define snmp_set_sysname(ocstr, ocstrlen);`

Definition at line 241 of file snmp.h.

5.76.1.113 `#define snmp_set_sysobjid(oid);`

Definition at line 235 of file snmp.h.

5.76.2 Enumeration Type Documentation

5.76.2.1 `enum snmp_ifType`

See also

RFC1213, "MIB-II, 6. Definitions"

Enumerator

snmp_ifType_other

snmp_ifType_regular1822

snmp_ifType_hdh1822

snmp_ifType_ddn_x25

snmp_ifType_rfc877_x25

snmp_ifType_ethernet_csmacd

snmp_ifType_iso88023_csmacd

snmp_ifType_iso88024_tokenBus

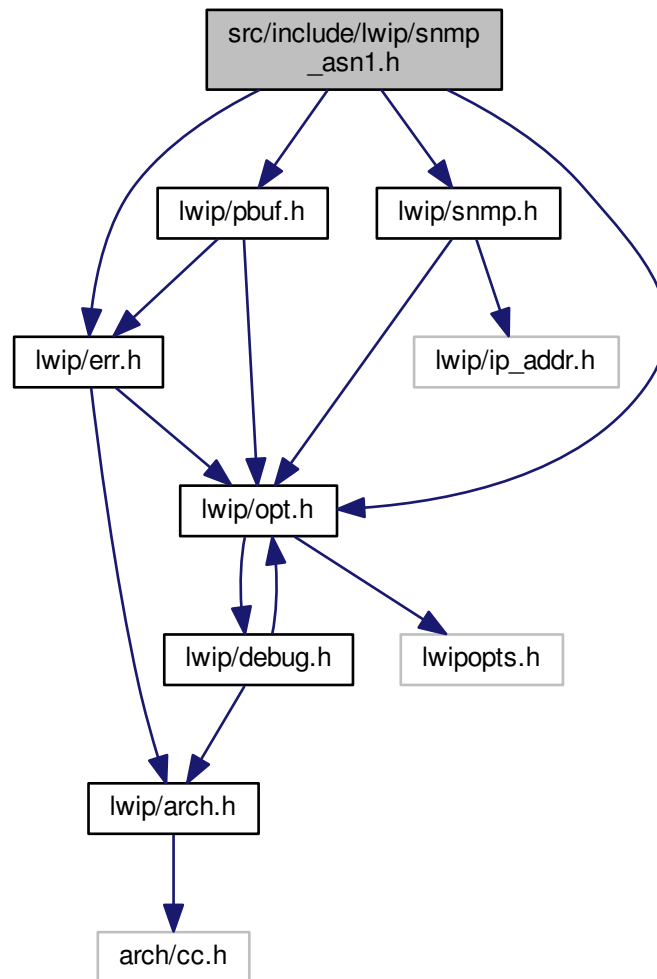
snmp_ifType_iso88025_tokenRing
snmp_ifType_iso88026_man
snmp_ifType_starLan
snmp_ifType_proteon_10Mbit
snmp_ifType_proteon_80Mbit
snmp_ifType_hyperchannel
snmp_ifType_fddi
snmp_ifType_lapb
snmp_ifType_sdlc
snmp_ifType_ds1
snmp_ifType_e1
snmp_ifType_basicISDN
snmp_ifType_primaryISDN
snmp_ifType_propPointToPointSerial
snmp_ifType_ppp
snmp_ifType_softwareLoopback
snmp_ifType_eon
snmp_ifType_ethernet_3Mbit
snmp_ifType_nsip
snmp_ifType_slip
snmp_ifType_ultra
snmp_ifType_ds3
snmp_ifType_sip
snmp_ifType_frame_relay

Definition at line 50 of file snmp.h.

5.77 src/include/lwip/snmp_asn1.h File Reference

```
#include "lwip/opt.h"  
#include "lwip/err.h"  
#include "lwip/pbuf.h"  
#include "lwip/snmp.h"
```

Include dependency graph for snmp_asn1.h:



5.77.1 Detailed Description

Abstract Syntax Notation One (ISO 8824, 8825) codec.

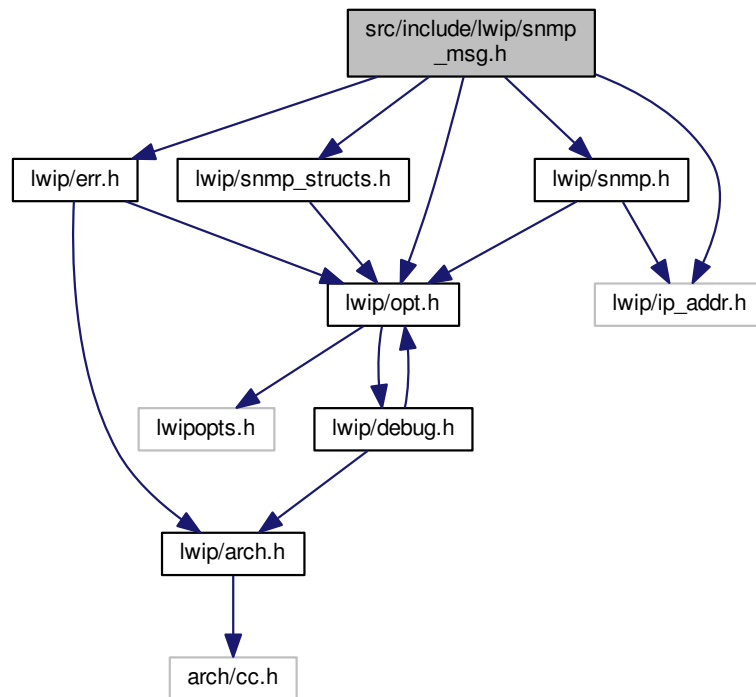
5.78 src/include/lwip/snmp_msg.h File Reference

```

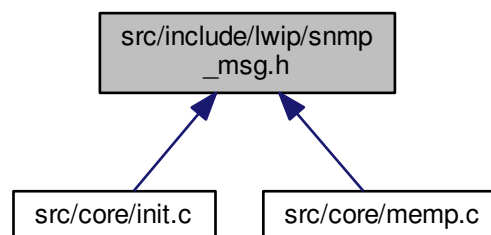
#include "lwip/opt.h"
#include "lwip/snmp.h"
#include "lwip/snmp_structs.h"
#include "lwip/ip_addr.h"
#include "lwip/err.h"

```

Include dependency graph for snmp_msg.h:



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



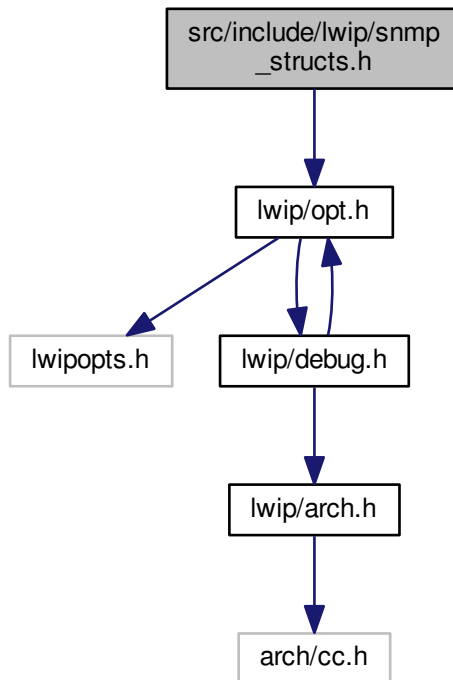
5.78.1 Detailed Description

SNMP Agent message handling structures.

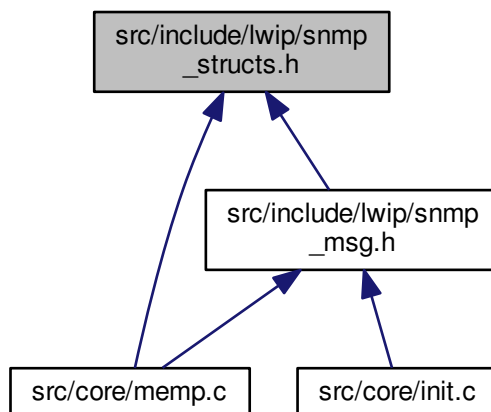
5.79 src/include/lwip/snmp_structs.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for snmp_structs.h:



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



5.79.1 Detailed Description

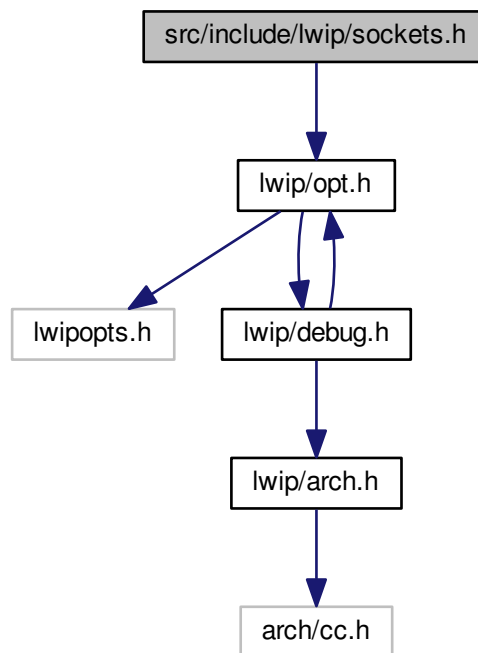
Generic MIB tree structures.

Todo namespace prefixes

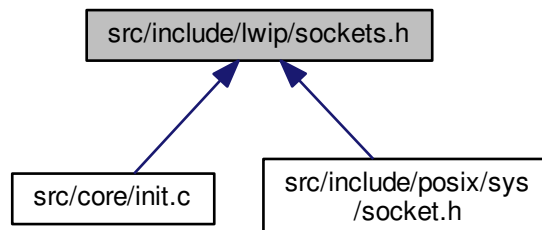
5.80 src/include/lwip/sockets.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for sockets.h:

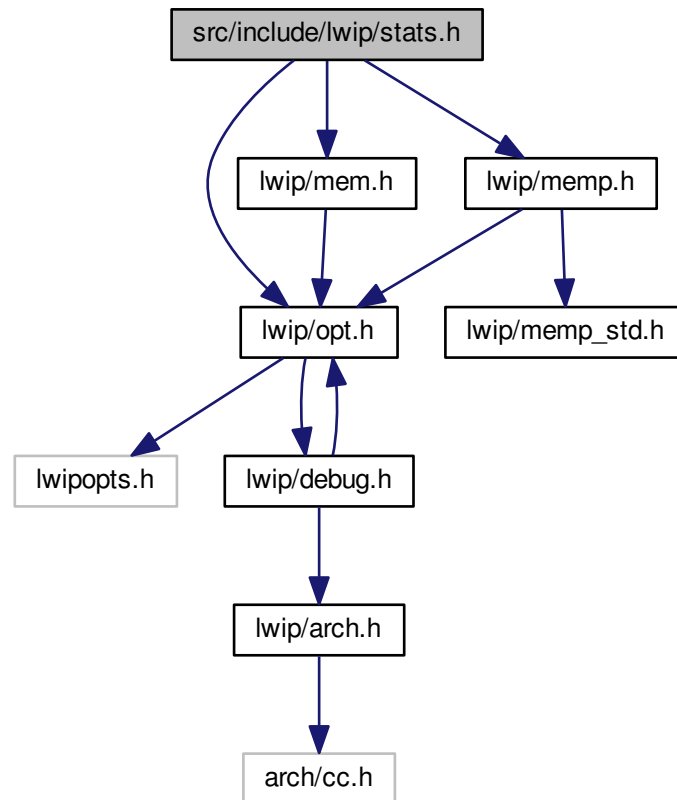


This graph shows which files directly or indirectly include this file:



5.81 src/include/lwip/stats.h File Reference

```
#include "lwip/opt.h"
#include "lwip/mem.h"
#include "lwip/memp.h"
Include dependency graph for stats.h:
```



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



Macros

- `#define stats_init()`
- `#define STATS_INC(x)`
- `#define STATS_DEC(x)`
- `#define STATS_INC_USED(x)`
- `#define TCP_STATS_INC(x)`

- `#define TCP_STATS_DISPLAY()`
- `#define UDP_STATS_INC(x)`
- `#define UDP_STATS_DISPLAY()`
- `#define ICMP_STATS_INC(x)`
- `#define ICMP_STATS_DISPLAY()`
- `#define IGMP_STATS_INC(x)`
- `#define IGMP_STATS_DISPLAY()`
- `#define IP_STATS_INC(x)`
- `#define IP_STATS_DISPLAY()`
- `#define IPFRAG_STATS_INC(x)`
- `#define IPFRAG_STATS_DISPLAY()`
- `#define ETHARP_STATS_INC(x)`
- `#define ETHARP_STATS_DISPLAY()`
- `#define LINK_STATS_INC(x)`
- `#define LINK_STATS_DISPLAY()`
- `#define MEM_STATS_AVAIL(x, y)`
- `#define MEM_STATS_INC(x)`
- `#define MEM_STATS_INC_USED(x, y)`
- `#define MEM_STATS_DEC_USED(x, y)`
- `#define MEM_STATS_DISPLAY()`
- `#define MEMP_STATS_AVAIL(x, i, y)`
- `#define MEMP_STATS_INC(x, i)`
- `#define MEMP_STATS_DEC(x, i)`
- `#define MEMP_STATS_INC_USED(x, i)`
- `#define MEMP_STATS_DISPLAY(i)`
- `#define SYS_STATS_INC(x)`
- `#define SYS_STATS_DEC(x)`
- `#define SYS_STATS_INC_USED(x)`
- `#define SYS_STATS_DISPLAY()`
- `#define stats_display()`
- `#define stats_display_proto(proto, name)`
- `#define stats_display_igmp(igmp)`
- `#define stats_display_mem(mem, name)`
- `#define stats_display_memp(mem, index)`
- `#define stats_display_sys(sys)`

5.81.1 Macro Definition Documentation

5.81.1.1 `#define ETHARP_STATS_DISPLAY()`

Definition at line 220 of file stats.h.

5.81.1.2 `#define ETHARP_STATS_INC(x)`

Definition at line 219 of file stats.h.

5.81.1.3 `#define ICMP_STATS_DISPLAY()`

Definition at line 188 of file stats.h.

5.81.1.4 `#define ICMP_STATS_INC(x)`

Definition at line 187 of file stats.h.

5.81.1.5 #define IGMP_STATS_DISPLAY()

Definition at line 196 of file stats.h.

5.81.1.6 #define IGMP_STATS_INC(x)

Definition at line 195 of file stats.h.

5.81.1.7 #define IP_STATS_DISPLAY()

Definition at line 204 of file stats.h.

5.81.1.8 #define IP_STATS_INC(x)

Definition at line 203 of file stats.h.

5.81.1.9 #define IPFRAG_STATS_DISPLAY()

Definition at line 212 of file stats.h.

5.81.1.10 #define IPFRAG_STATS_INC(x)

Definition at line 211 of file stats.h.

5.81.1.11 #define LINK_STATS_DISPLAY()

Definition at line 228 of file stats.h.

5.81.1.12 #define LINK_STATS_INC(x)

Definition at line 227 of file stats.h.

5.81.1.13 #define MEM_STATS_AVAIL(x, y)

Definition at line 238 of file stats.h.

5.81.1.14 #define MEM_STATS_DEC_USED(x, y)

Definition at line 241 of file stats.h.

5.81.1.15 #define MEM_STATS_DISPLAY()

Definition at line 242 of file stats.h.

5.81.1.16 #define MEM_STATS_INC(x)

Definition at line 239 of file stats.h.

5.81.1.17 `#define MEM_STATS_INC_USED(x, y)`

Definition at line 240 of file stats.h.

5.81.1.18 `#define MEMP_STATS_AVAIL(x, i, y)`

Definition at line 252 of file stats.h.

5.81.1.19 `#define MEMP_STATS_DEC(x, i)`

Definition at line 254 of file stats.h.

5.81.1.20 `#define MEMP_STATS_DISPLAY(i)`

Definition at line 256 of file stats.h.

5.81.1.21 `#define MEMP_STATS_INC(x, i)`

Definition at line 253 of file stats.h.

5.81.1.22 `#define MEMP_STATS_INC_USED(x, i)`

Definition at line 255 of file stats.h.

5.81.1.23 `#define STATS_DEC(x)`

Definition at line 163 of file stats.h.

5.81.1.24 `#define stats_display()`

Definition at line 280 of file stats.h.

5.81.1.25 `#define stats_display_igmp(igmp)`

Definition at line 282 of file stats.h.

5.81.1.26 `#define stats_display_mem(mem, name)`

Definition at line 283 of file stats.h.

5.81.1.27 `#define stats_display_memp(mem, index)`

Definition at line 284 of file stats.h.

5.81.1.28 `#define stats_display_proto(proto, name)`

Definition at line 281 of file stats.h.

5.81.1.29 `#define stats_display_sys(sys)`

Definition at line 285 of file stats.h.

5.81.1.30 `#define STATS_INC(x)`

Definition at line 162 of file stats.h.

5.81.1.31 `#define STATS_INC_USED(x)`

Definition at line 164 of file stats.h.

5.81.1.32 `#define stats_init()`

Definition at line 161 of file stats.h.

5.81.1.33 `#define SYS_STATS_DEC(x)`

Definition at line 266 of file stats.h.

5.81.1.34 `#define SYS_STATS_DISPLAY()`

Definition at line 268 of file stats.h.

5.81.1.35 `#define SYS_STATS_INC(x)`

Definition at line 265 of file stats.h.

5.81.1.36 `#define SYS_STATS_INC_USED(x)`

Definition at line 267 of file stats.h.

5.81.1.37 `#define TCP_STATS_DISPLAY()`

Definition at line 172 of file stats.h.

5.81.1.38 `#define TCP_STATS_INC(x)`

Definition at line 171 of file stats.h.

5.81.1.39 `#define UDP_STATS_DISPLAY()`

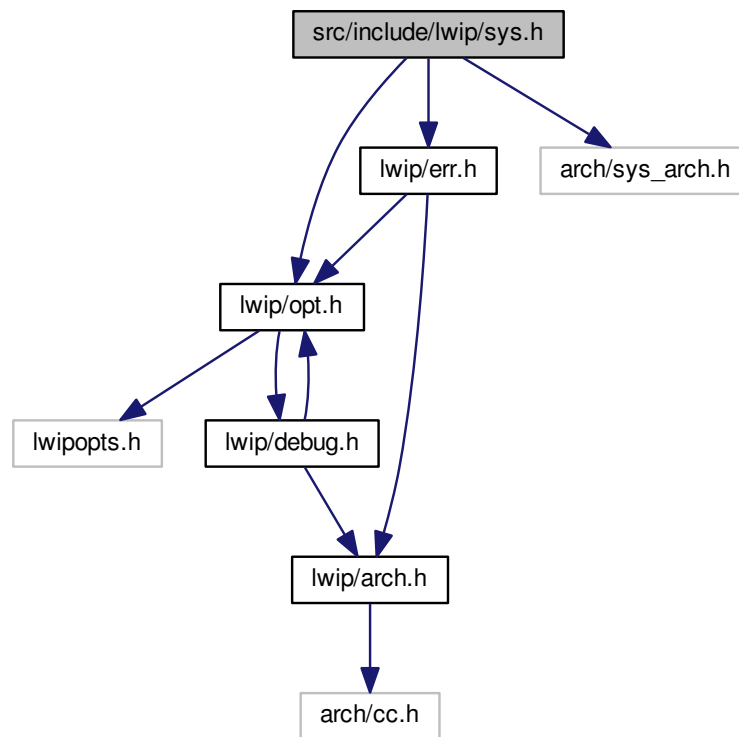
Definition at line 180 of file stats.h.

5.81.1.40 `#define UDP_STATS_INC(x)`

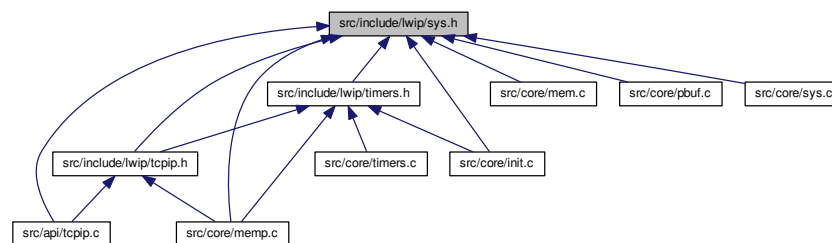
Definition at line 179 of file stats.h.

5.82 src/include/lwip/sys.h File Reference

```
#include "lwip/opt.h"
#include "lwip/err.h"
#include "arch/sys_arch.h"
Include dependency graph for sys.h:
```



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



Macros

- `#define SYS_ARCH_TIMEOUT 0xffffffffUL`
- `#define SYS_MBOX_EMPTY SYS_ARCH_TIMEOUT`

- `#define sys_sem_wait(sem) sys_arch_sem_wait(sem, 0)`
- `#define sys_mbox_tryfetch(mbox, msg) sys_arch_mbox_tryfetch(mbox, msg)`
- `#define sys_mbox_fetch(mbox, msg) sys_arch_mbox_fetch(mbox, msg, 0)`
- `#define SYS_ARCH_DECL_PROTECT(lev)`
- `#define SYS_ARCH_PROTECT(lev)`
- `#define SYS_ARCH_UNPROTECT(lev)`
- `#define SYS_ARCH_INC(var, val)`
- `#define SYS_ARCH_DEC(var, val)`
- `#define SYS_ARCH_GET(var, ret)`
- `#define SYS_ARCH_SET(var, val)`

Typedefs

- `typedef void(* lwip_thread_fn) (void *arg)`

Functions

- `err_t sys_mutex_new (sys_mutex_t *mutex)`
- `void sys_mutex_lock (sys_mutex_t *mutex)`
- `void sys_mutex_unlock (sys_mutex_t *mutex)`
- `void sys_mutex_free (sys_mutex_t *mutex)`
- `int sys_mutex_valid (sys_mutex_t *mutex)`
- `void sys_mutex_set_invalid (sys_mutex_t *mutex)`
- `err_t sys_sem_new (sys_sem_t *sem, u8_t count)`
- `void sys_sem_signal (sys_sem_t *sem)`
- `u32_t sys_arch_sem_wait (sys_sem_t *sem, u32_t timeout)`
- `void sys_sem_free (sys_sem_t *sem)`
- `int sys_sem_valid (sys_sem_t *sem)`
- `void sys_sem_set_invalid (sys_sem_t *sem)`
- `void sys_msleep (u32_t ms)`
- `err_t sys_mbox_new (sys_mbox_t *mbox, int size)`
- `void sys_mbox_post (sys_mbox_t *mbox, void *msg)`
- `err_t sys_mbox_trypost (sys_mbox_t *mbox, void *msg)`
- `u32_t sys_arch_mbox_fetch (sys_mbox_t *mbox, void **msg, u32_t timeout)`
- `u32_t sys_arch_mbox_tryfetch (sys_mbox_t *mbox, void **msg)`
- `void sys_mbox_free (sys_mbox_t *mbox)`
- `int sys_mbox_valid (sys_mbox_t *mbox)`
- `void sys_mbox_set_invalid (sys_mbox_t *mbox)`
- `sys_thread_t sys_thread_new (const char *name, lwip_thread_fn thread, void *arg, int stacksize, int prio)`
- `void sys_init (void)`
- `u32_t sys_jiffies (void)`
- `u32_t sys_now (void)`

5.82.1 Macro Definition Documentation

5.82.1.1 `#define SYS_ARCH_DEC(var, val)`

Value:

```
do { \
    old_level); \
    SYS_ARCH_DECL_PROTECT( \
    SYS_ARCH_PROTECT(old_level) \
    ; \
    var -= val; \
    SYS_ARCH_UNPROTECT( \
    old_level); \
} while(0)
```

Definition at line 306 of file sys.h.

5.82.1.2 `#define SYS_ARCH_DECL_PROTECT(lev)`

`SYS_LIGHTWEIGHT_PROT` define `SYS_LIGHTWEIGHT_PROT` in `lwipopts.h` if you want inter-task protection for certain critical regions during buffer allocation, deallocation and memory allocation and deallocation.

Definition at line 283 of file sys.h.

5.82.1.3 `#define SYS_ARCH_GET(var, ret)`

Value:

```
do { \
    old_level); \
    ; \
    old_level); \
    } while(0)
```

Definition at line 315 of file sys.h.

5.82.1.4 `#define SYS_ARCH_INC(var, val)`

Value:

```
do { \
    old_level); \
    ; \
    old_level); \
    } while(0)
```

Definition at line 297 of file sys.h.

5.82.1.5 `#define SYS_ARCH_PROTECT(lev)`

Definition at line 284 of file sys.h.

5.82.1.6 `#define SYS_ARCH_SET(var, val)`

Value:

```
do { \
    old_level); \
    ; \
    old_level); \
    } while(0)
```

Definition at line 324 of file sys.h.

5.82.1.7 #define SYS_ARCH_TIMEOUT 0xffffffffUL

Return code for timeouts from `sys_arch_mbox_fetch` and `sys_arch_sem_wait`

Definition at line 78 of file `sys.h`.

5.82.1.8 #define SYS_ARCH_UNPROTECT(lev)

Definition at line 285 of file `sys.h`.

5.82.1.9 #define SYS_MBOX_EMPTY SYS_ARCH_TIMEOUT

`sys_mbox_tryfetch()` returns `SYS_MBOX_EMPTY` if appropriate. For now we use the same magic value, but we allow this to change in future.

Definition at line 83 of file `sys.h`.

5.82.1.10 #define sys_mbox_fetch(mbox, msg) sys_arch_mbox_fetch(mbox, msg, 0)

Definition at line 207 of file `sys.h`.

5.82.1.11 #define sys_mbox_tryfetch(mbox, msg) sys_arch_mbox_tryfetch(mbox, msg)

For now, we map straight to `sys_arch` implementation.

Definition at line 203 of file `sys.h`.

5.82.1.12 #define sys_sem_wait(sem) sys_arch_sem_wait(sem, 0)

Wait for a semaphore - forever/no timeout

Definition at line 153 of file `sys.h`.

5.82.2 Typedef Documentation**5.82.2.1 typedef void(* lwip_thread_fn) (void *arg)**

Function prototype for thread functions

Definition at line 89 of file `sys.h`.

5.82.3 Function Documentation**5.82.3.1 u32_t sys_arch_mbox_fetch (sys_mbox_t * mbox, void ** msg, u32_t timeout)**

Wait for a new message to arrive in the mbox

Parameters

<i>mbox</i>	mbox to get a message from
<i>msg</i>	pointer where the message is stored

<i>timeout</i>	maximum time (in milliseconds) to wait for a message
----------------	--

Returns

time (in milliseconds) waited for a message, may be 0 if not waited or SYS_ARCH_TIMEOUT on timeout The returned time has to be accurate to prevent timer jitter!

5.82.3.2 u32_t sys_arch_mbox_tryfetch (sys_mbox_t * *mbox*, void ** *msg*)

Wait for a new message to arrive in the mbox

Parameters

<i>mbox</i>	mbox to get a message from
<i>msg</i>	pointer where the message is stored
<i>timeout</i>	maximum time (in milliseconds) to wait for a message

Returns

0 (milliseconds) if a message has been received or SYS_MBOX_EMPTY if the mailbox is empty

5.82.3.3 u32_t sys_arch_sem_wait (sys_sem_t * *sem*, u32_t *timeout*)

Wait for a semaphore for the specified timeout

Parameters

<i>sem</i>	the semaphore to wait for
<i>timeout</i>	timeout in milliseconds to wait (0 = wait forever)

Returns

time (in milliseconds) waited for the semaphore or SYS_ARCH_TIMEOUT on timeout

5.82.3.4 void sys_init (void)

5.82.3.5 u32_t sys_jiffies (void)

Ticks/jiffies since power up.

5.82.3.6 void sys_mbox_free (sys_mbox_t * *mbox*)

Delete an mbox

Parameters

<i>mbox</i>	mbox to delete
-------------	----------------

5.82.3.7 err_t sys_mbox_new (sys_mbox_t * *mbox*, int *size*)

Create a new mbox of specified size

Parameters

<i>mbox</i>	pointer to the mbox to create
<i>size</i>	(miminum) number of messages in this mbox

Returns

ERR_OK if successful, another err_t otherwise

5.82.3.8 void sys_mbox_post (sys_mbox_t * *mbox*, void * *msg*)

Post a message to an mbox - may not fail -> blocks if full, only used from tasks not from ISR

Parameters

<i>mbox</i>	mbox to posts the message
<i>msg</i>	message to post (ATTENTION: can be NULL)

5.82.3.9 void sys_mbox_set_invalid (sys_mbox_t * *mbox*)

Set an mbox invalid so that sys_mbox_valid returns 0

5.82.3.10 err_t sys_mbox_trypost (sys_mbox_t * *mbox*, void * *msg*)

Try to post a message to an mbox - may fail if full or ISR

Parameters

<i>mbox</i>	mbox to posts the message
<i>msg</i>	message to post (ATTENTION: can be NULL)

5.82.3.11 int sys_mbox_valid (sys_mbox_t * *mbox*)

Check if an mbox is valid/allocated: return 1 for valid, 0 for invalid

5.82.3.12 void sys_msleep (u32_t *ms*)

Sleep for some ms. Timeouts are NOT processed while sleeping.

Parameters

<i>ms</i>	number of milliseconds to sleep
-----------	---------------------------------

Definition at line 55 of file sys.c.

5.82.3.13 void sys_mutex_free (sys_mutex_t * *mutex*)

Delete a semaphore

Parameters

<i>mutex</i>	the mutex to delete
--------------	---------------------

5.82.3.14 void sys_mutex_lock (sys_mutex_t * *mutex*)

Lock a mutex

Parameters

<i>mutex</i>	the mutex to lock
--------------	-------------------

5.82.3.15 err_t sys_mutex_new (sys_mutex_t * *mutex*)

Define LWIP_COMPAT_MUTEX if the port has no mutexes and binary semaphores should be used instead Create a new mutex

Parameters

<i>mutex</i>	pointer to the mutex to create
--------------	--------------------------------

Returns

a new mutex

5.82.3.16 void sys_mutex_set_invalid (sys_mutex_t * *mutex*)

Set a mutex invalid so that sys_mutex_valid returns 0

5.82.3.17 void sys_mutex_unlock (sys_mutex_t * *mutex*)

Unlock a mutex

Parameters

<i>mutex</i>	the mutex to unlock
--------------	---------------------

5.82.3.18 int sys_mutex_valid (sys_mutex_t * *mutex*)

Check if a mutex is valid/allocated: return 1 for valid, 0 for invalid

5.82.3.19 u32_t sys_now (void)

Returns the current time in milliseconds, may be the same as sys_jiffies or at least based on it.

5.82.3.20 void sys_sem_free (sys_sem_t * *sem*)

Delete a semaphore

Parameters

<i>sem</i>	semaphore to delete
------------	---------------------

5.82.3.21 `err_t sys_sem_new (sys_sem_t * sem, u8_t count)`

Create a new semaphore

Parameters

<i>sem</i>	pointer to the semaphore to create
<i>count</i>	initial count of the semaphore

Returns

ERR_OK if successful, another err_t otherwise

5.82.3.22 `void sys_sem_set_invalid (sys_sem_t * sem)`

Set a semaphore invalid so that sys_sem_valid returns 0

5.82.3.23 `void sys_sem_signal (sys_sem_t * sem)`

Signals a semaphore

Parameters

<i>sem</i>	the semaphore to signal
------------	-------------------------

5.82.3.24 `int sys_sem_valid (sys_sem_t * sem)`

Check if a semaphore is valid/allocated: return 1 for valid, 0 for invalid

5.82.3.25 `sys_thread_t sys_thread_new (const char * name, lwip_thread_fn thread, void * arg, int stacksize, int prio)`

The only thread function: Creates a new thread

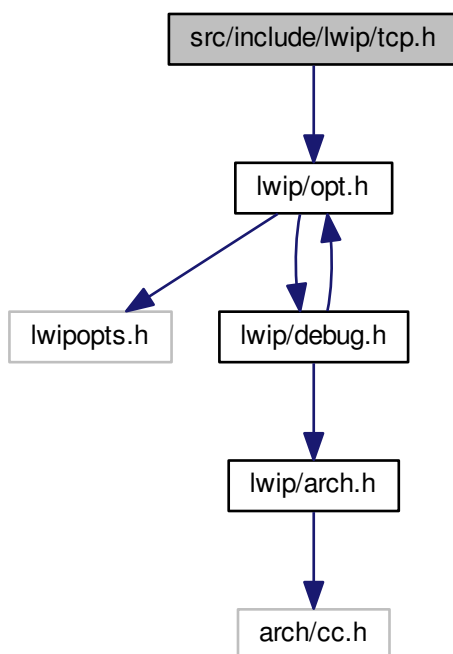
Parameters

<i>name</i>	human-readable name for the thread (used for debugging purposes)
<i>thread</i>	thread-function
<i>arg</i>	parameter passed to 'thread'
<i>stacksize</i>	stack size in bytes for the new thread (may be ignored by ports)
<i>prio</i>	priority of the new thread (may be ignored by ports)

5.83 src/include/lwip/tcp.h File Reference

```
#include "lwip/opt.h"
```

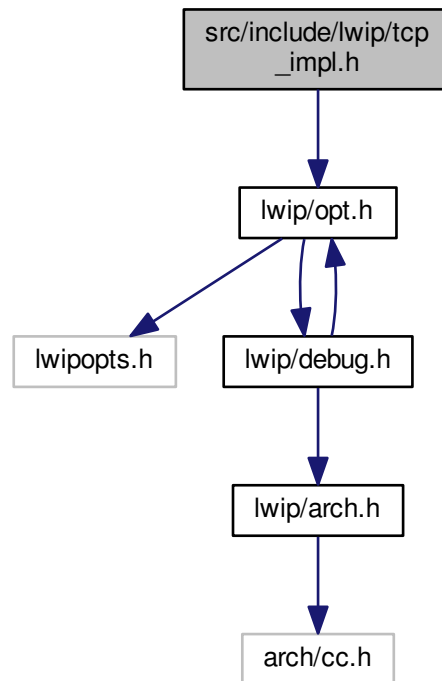
Include dependency graph for tcp.h:



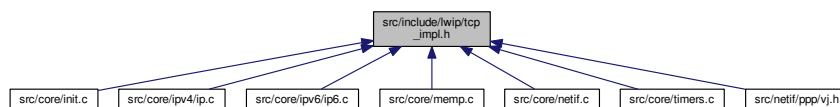
5.84 src/include/lwip/tcp_impl.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for tcp_impl.h:



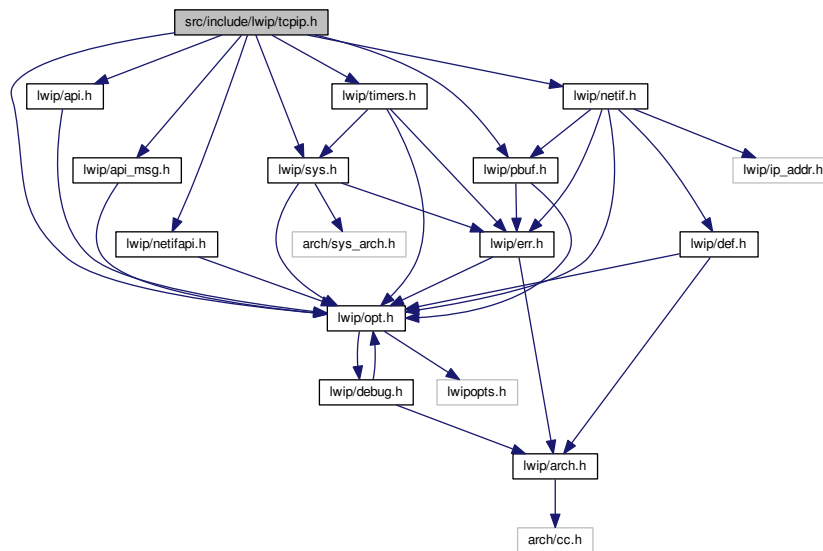
This graph shows which files directly or indirectly include this file:



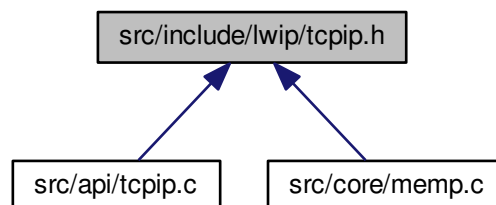
5.85 src/include/lwip/tcpip.h File Reference

```
#include "lwip/opt.h"
#include "lwip/api_msg.h"
#include "lwip/netifapi.h"
#include "lwip/pbuf.h"
#include "lwip/api.h"
#include "lwip/sys.h"
#include "lwip/timers.h"
#include "lwip/netif.h"
```

Include dependency graph for tcpip.h:



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



Data Structures

- struct [tcpip_msg](#)

Macros

- `#define LWIP_TCPIP_THREAD_ALIVE()`
- `#define LOCK_TCPIP_CORE()`
- `#define UNLOCK_TCPIP_CORE()`
- `#define TCPIP_APIMSG(m) tcpip_apimsg(m)`
- `#define TCPIP_APIMSG_ACK(m) sys_sem_signal(&m->conn->op_completed)`
- `#define TCPIP_NETIFAPI(m) tcpip_netifapi(m)`
- `#define TCPIP_NETIFAPI_ACK(m) sys_sem_signal(&m->sem)`
- `#define tcpip_callback(f, ctx) tcpip_callback_with_block(f, ctx, 1)`

Typedefs

- typedef void(* [tcpip_init_done_fn](#)) (void *arg)
- typedef void(* [tcpip_callback_fn](#)) (void *ctx)

Enumerations

- enum [tcpip_msg_type](#) { [TCPIP_MSG_INPKT](#), [TCPIP_MSG_CALLBACK](#), [TCPIP_MSG_CALLBACK_STATUS_TIC](#) }

Functions

- void [tcpip_init](#) ([tcpip_init_done_fn](#) tcpip_init_done, void *arg)
- [err_t](#) [tcpip_input](#) (struct [pbuf](#) *p, struct [netif](#) *inp)
- [err_t](#) [tcpip_callback_with_block](#) ([tcpip_callback_fn](#) function, void *ctx, u8_t block)
- struct [tcpip_callback_msg](#) * [tcpip_callbackmsg_new](#) ([tcpip_callback_fn](#) function, void *ctx)
- void [tcpip_callbackmsg_delete](#) (struct [tcpip_callback_msg](#) *msg)
- [err_t](#) [tcpip_trycallback](#) (struct [tcpip_callback_msg](#) *msg)
- [err_t](#) [pbuf_free_callback](#) (struct [pbuf](#) *p)
- [err_t](#) [mem_free_callback](#) (void *m)

5.85.1 Macro Definition Documentation

5.85.1.1 #define LOCK_TCPIP_CORE()

Definition at line 67 of file [tcpip.h](#).

5.85.1.2 #define LWIP_TCPIP_THREAD_ALIVE()

Define this to something that triggers a watchdog. This is called from [tcpip_thread](#) after processing a message.

Definition at line 54 of file [tcpip.h](#).

5.85.1.3 #define TCPIP_APIMSG(m) tcpip_apimsg(m)

Definition at line 69 of file [tcpip.h](#).

5.85.1.4 #define TCPIP_APIMSG_ACK(m) sys_sem_signal(&m->conn->op_completed)

Definition at line 70 of file [tcpip.h](#).

5.85.1.5 #define tcpip_callback(f, ctx) tcpip_callback_with_block(f, ctx, 1)

Definition at line 102 of file [tcpip.h](#).

5.85.1.6 #define TCPIP_NETIFAPI(m) tcpip_netifapi(m)

Definition at line 71 of file [tcpip.h](#).

5.85.1.7 #define TCPIP_NETIFAPI_ACK(m) sys_sem_signal(&m->sem)

Definition at line 72 of file [tcpip.h](#).

5.85.1.8 `#define UNLOCK_TCPIP_CORE()`

Definition at line 68 of file tcpip.h.

5.85.2 Typedef Documentation

5.85.2.1 `typedef void(* tcpip_callback_fn)(void *ctx)`

Function prototype for functions passed to [tcpip_callback\(\)](#)

Definition at line 78 of file tcpip.h.

5.85.2.2 `typedef void(* tcpip_init_done_fn)(void *arg)`

Function prototype for the init_done function passed to tcpip_init

Definition at line 76 of file tcpip.h.

5.85.3 Enumeration Type Documentation

5.85.3.1 `enum tcpip_msg_type`

Enumerator

TCPIP_MSG_INPKT

TCPIP_MSG_CALLBACK

TCPIP_MSG_CALLBACK_STATIC

Definition at line 117 of file tcpip.h.

5.85.4 Function Documentation

5.85.4.1 `err_t mem_free_callback(void * m)`

A simple wrapper function that allows you to free heap memory from interrupt context.

Parameters

<i>m</i>	the heap memory to free
----------	-------------------------

Returns

ERR_OK if callback could be enqueued, an err_t if not

Definition at line 506 of file tcpip.c.

5.85.4.2 `err_t pbuf_free_callback(struct pbuf * p)`

A simple wrapper function that allows you to free a pbuf from interrupt context.

Parameters

<i>p</i>	The pbuf (chain) to be dereferenced.
----------	--------------------------------------

Returns

ERR_OK if callback could be enqueued, an err_t if not

Definition at line 493 of file tcpip.c.

5.85.4.3 err_t tcpip_callback_with_block (tcpip_callback_fn function, void * ctx, u8_t block)

Call a specific function in the thread context of tcpip_thread for easy access synchronization. A function called in that way may access lwIP core code without fearing concurrent access.

Parameters

<i>f</i>	the function to call
<i>ctx</i>	parameter passed to f
<i>block</i>	1 to block until the request is posted, 0 to non-blocking mode

Returns

ERR_OK if the function was called, another err_t if not

Definition at line 211 of file tcpip.c.

5.85.4.4 void tcpip_callbackmsg_delete (struct tcpip_callback_msg * msg)

Free a callback message allocated by [tcpip_callbackmsg_new\(\)](#).

Parameters

<i>msg</i>	the message to free
------------	---------------------

Definition at line 425 of file tcpip.c.

5.85.4.5 struct tcpip_callback_msg* tcpip_callbackmsg_new (tcpip_callback_fn function, void * ctx)

Allocate a structure for a static callback message and initialize it. This is intended to be used to send "static" messages from interrupt context.

Parameters

<i>function</i>	the function to call
<i>ctx</i>	parameter passed to function

Returns

a struct pointer to pass to [tcpip_trycallback\(\)](#).

Definition at line 408 of file tcpip.c.

5.85.4.6 void tcpip_init (tcpip_init_done_fn initfunc, void * arg)

Initialize this module:

- initialize all sub modules
- start the tcpip_thread

Parameters

<i>initfunc</i>	a function to call when tcpip_thread is running and finished initializing
<i>arg</i>	argument to pass to initfunc

Definition at line 455 of file tcpip.c.

5.85.4.7 `err_t tcpip_input (struct pbuf * p, struct netif * inp)`

Pass a received packet to tcpip_thread for input processing

Parameters

<i>p</i>	the received packet, p->payload pointing to the Ethernet header or to an IP header (if inp doesn't have NETIF_FLAG_ETHARP or NETIF_FLAG_ETHERNET flags)
<i>inp</i>	the network interface on which the packet was received

Definition at line 161 of file tcpip.c.

5.85.4.8 `err_t tcpip_trycallback (struct tcpip_callback_msg * msg)`

Try to post a callback-message to the tcpip_thread mbox This is intended to be used to send "static" messages from interrupt context.

Parameters

<i>msg</i>	pointer to the message to post
------------	--------------------------------

Returns

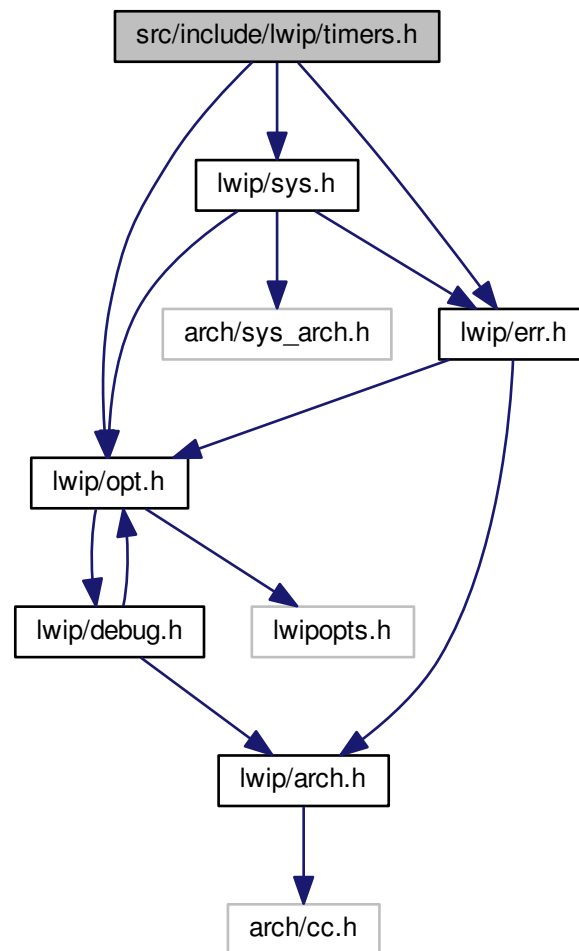
[sys_mbox_trypost\(\)](#) return code

Definition at line 438 of file tcpip.c.

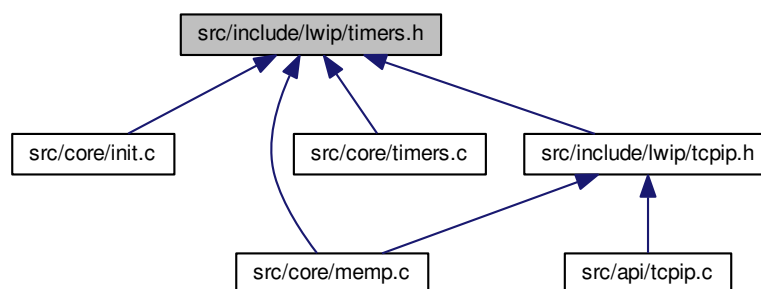
5.86 `src/include/lwip/timers.h` File Reference

```
#include "lwip/opt.h"
#include "lwip/err.h"
#include "lwip/sys.h"
```

Include dependency graph for timers.h:



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



Data Structures

- struct [sys_timeo](#)

Macros

- `#define LWIP_TIMERS (!NO_SYS || (NO_SYS && !NO_SYS_NO_TIMERS))`
- `#define LWIP_DEBUG_TIMERNAMES 0`

Typedefs

- `typedef void(* sys_timeout_handler) (void *arg)`

Functions

- void [sys_timeouts_init](#) (void)
- void [sys_timeout](#) (u32_t msec, [sys_timeout_handler](#) handler, void *arg)
- void [sys_untimeout](#) ([sys_timeout_handler](#) handler, void *arg)
- void [sys_timeouts_mbox_fetch](#) (sys_mbox_t *mbox, void **msg)

5.86.1 Macro Definition Documentation

5.86.1.1 `#define LWIP_DEBUG_TIMERNAMES 0`

Definition at line 56 of file timers.h.

5.86.1.2 `#define LWIP_TIMERS (!NO_SYS || (NO_SYS && !NO_SYS_NO_TIMERS))`

Definition at line 39 of file timers.h.

5.86.2 Typedef Documentation

5.86.2.1 `typedef void(* sys_timeout_handler) (void *arg)`

Function prototype for a timeout callback function. Register such a function using [sys_timeout\(\)](#).

Parameters

<i>arg</i>	Additional argument to pass to the function - set up by sys_timeout()
------------	---

Definition at line 65 of file timers.h.

5.86.3 Function Documentation

5.86.3.1 `void sys_timeout (u32_t msec, sys_timeout_handler handler, void * arg)`

5.86.3.2 `void sys_timeouts_init (void)`

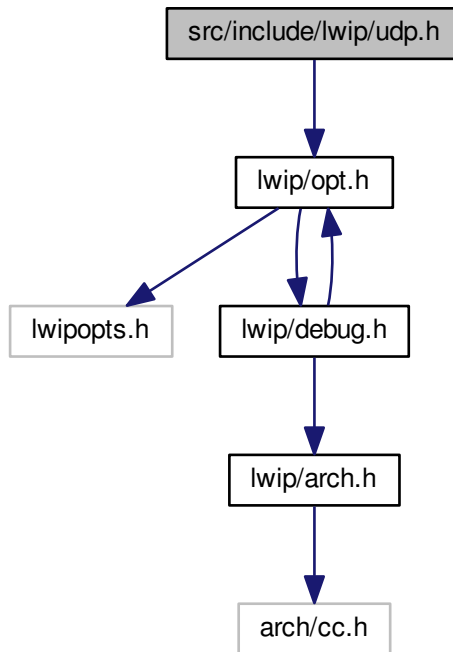
5.86.3.3 `void sys_timeouts_mbox_fetch (sys_mbox_t * mbox, void ** msg)`

5.86.3.4 `void sys_untimeout (sys_timeout_handler handler, void * arg)`

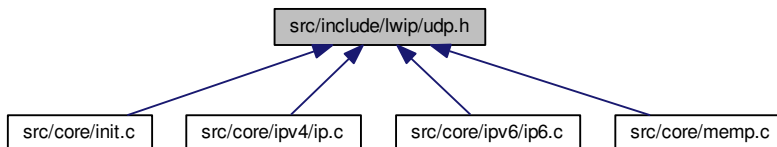
5.87 src/include/lwip/udp.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for udp.h:



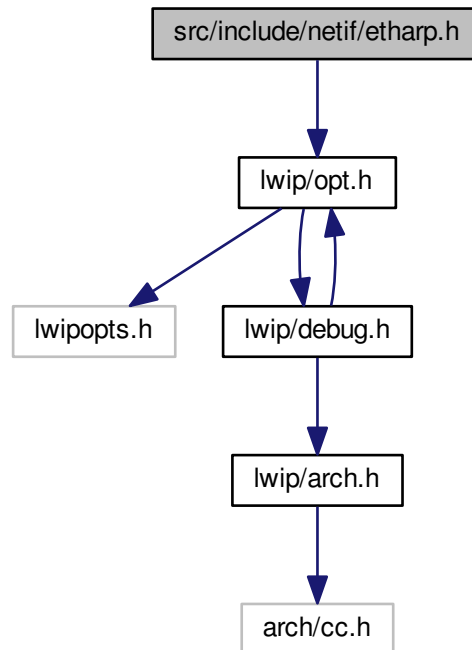
This graph shows which files directly or indirectly include this file:



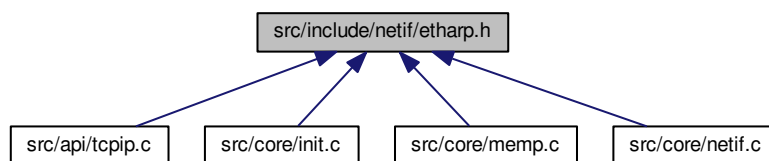
5.88 src/include/netif/etharp.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for etharp.h:



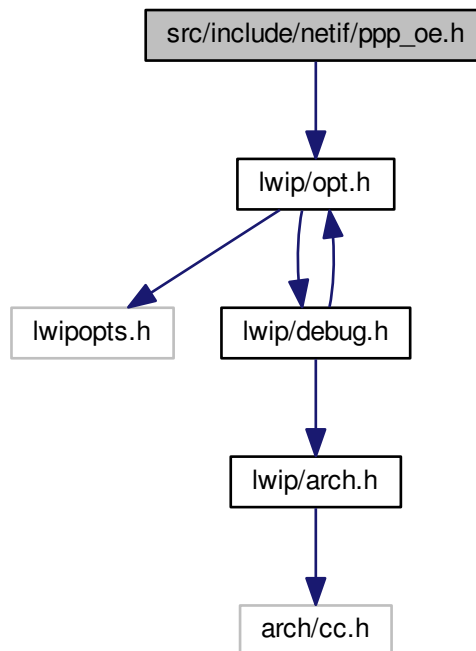
This graph shows which files directly or indirectly include this file:



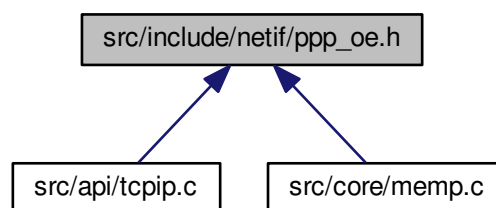
5.89 src/include/netif/ppp_oe.h File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for ppp_oe.h:



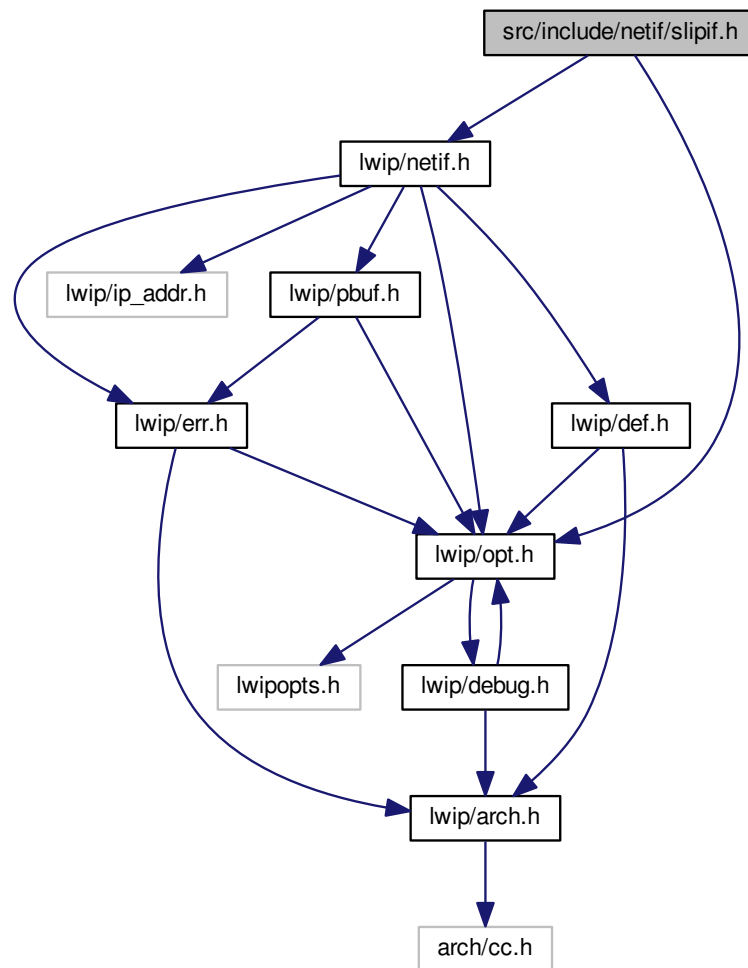
This graph shows which files directly or indirectly include this file:



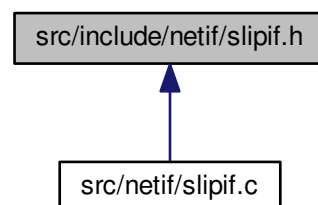
5.90 src/include/netif/sliph.h File Reference

```
#include "lwip/opt.h"
#include "lwip/netif.h"
```

Include dependency graph for `slipif.h`:



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



Macros

- `#define SLIP_USE_RX_THREAD !NO_SYS`
- `#define SLIP_RX_FROM_ISR 0`
- `#define SLIP_RX_QUEUE SLIP_RX_FROM_ISR`

Functions

- `err_t slipif_init (struct netif *netif)`
- `void slipif_poll (struct netif *netif)`

5.90.1 Macro Definition Documentation

5.90.1.1 `#define SLIP_RX_FROM_ISR 0`

Set this to 1 to enable functions to pass in RX bytes from ISR context. If enabled, `slipif_received_byte[s]()` process incoming bytes and put assembled packets on a queue, which is fed into lwIP from `slipif_poll()`. If disabled, `slipif_poll()` polls the serial line (using `sio_tryread()`).

Definition at line 53 of file `slipif.h`.

5.90.1.2 `#define SLIP_RX_QUEUE SLIP_RX_FROM_ISR`

Set this to 1 (default for `SLIP_RX_FROM_ISR`) to queue incoming packets received by `slipif_received_byte[s]()` as long as `PBUF_POOL` pbufs are available. If disabled, packets will be dropped if more than one packet is received.

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

5.90.1.3 `#define SLIP_USE_RX_THREAD !NO_SYS`

Set this to 1 to start a thread that blocks reading on the serial line (using `sio_read()`).

Definition at line 44 of file `slipif.h`.

5.90.2 Function Documentation

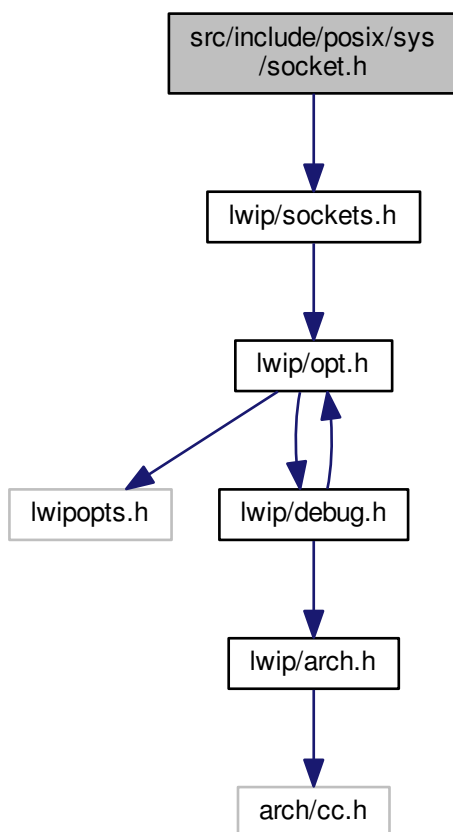
5.90.2.1 `err_t slipif_init (struct netif * netif)`

5.90.2.2 `void slipif_poll (struct netif * netif)`

5.91 src/include/posix/sys/socket.h File Reference

```
#include "lwip/sockets.h"
```

Include dependency graph for `socket.h`:



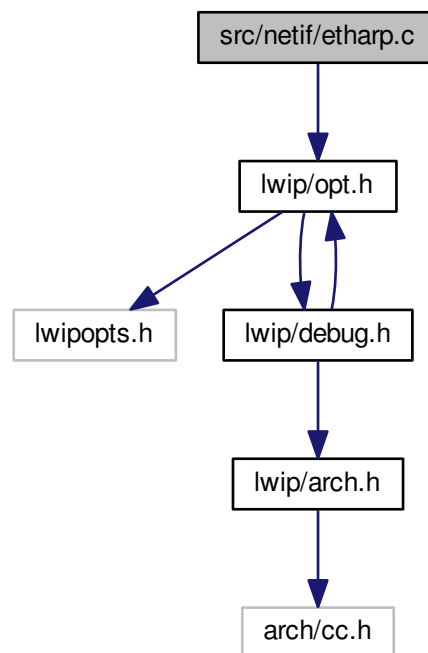
5.91.1 Detailed Description

This file is a posix wrapper for [lwip/sockets.h](#).

5.92 src/netif/etharp.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for etharp.c:



5.92.1 Detailed Description

Address Resolution Protocol module for IP over Ethernet

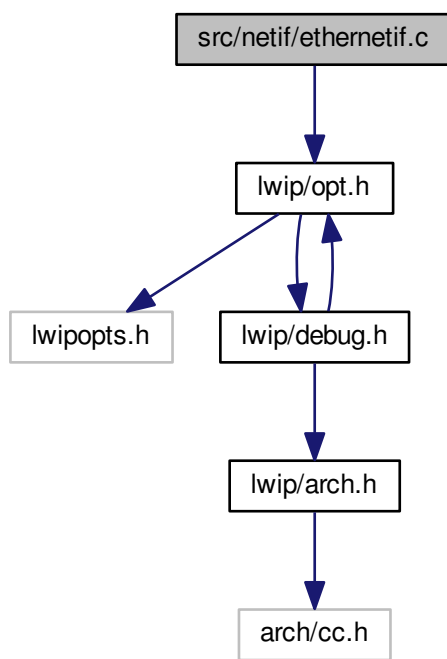
Functionally, ARP is divided into two parts. The first maps an IP address to a physical address when sending a packet, and the second part answers requests from other machines for our physical address.

This implementation complies with RFC 826 (Ethernet ARP). It supports Gratuitous ARP from RFC3220 (IP Mobility Support for IPv4) section 4.6 if an interface calls `etharp_gratuitous(our_netif)` upon address change.

5.93 src/netif/ethernetif.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for ethernetif.c:



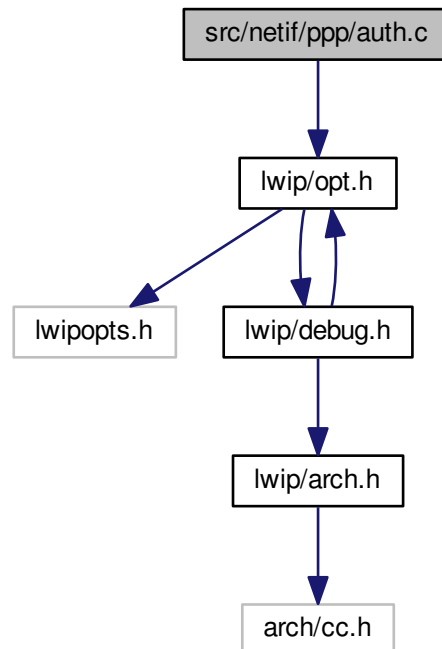
5.93.1 Detailed Description

Ethernet Interface Skeleton

5.94 src/netif/ppp/auth.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for auth.c:



5.95 src/netif/ppp/auth.h File Reference

Functions

- void `link_required` (int)
- void `link_terminated` (int)
- void `link_down` (int)
- void `link_established` (int)
- void `np_up` (int, u16_t)
- void `np_down` (int, u16_t)
- void `np_finished` (int, u16_t)
- void `auth_peer_fail` (int, u16_t)
- void `auth_peer_success` (int, u16_t, char *, int)
- void `auth_withpeer_fail` (int, u16_t)
- void `auth_withpeer_success` (int, u16_t)
- void `auth_check_options` (void)
- void `auth_reset` (int)
- u_char `check_passwd` (int, char *, int, char *, int, char **, int *)
- int `get_secret` (int, char *, char *, char *, int *, int)
- int `auth_ip_addr` (int, u32_t)
- int `bad_ip_adrs` (u32_t)

5.95.1 Function Documentation

5.95.1.1 void auth_check_options (void)

5.95.1.2 int auth_ip_addr (int, u32_t)

5.95.1.3 void auth_peer_fail (int, u16_t)

5.95.1.4 void auth_peer_success (int, u16_t, char *, int)

5.95.1.5 void auth_reset (int)

5.95.1.6 void auth_withpeer_fail (int, u16_t)

5.95.1.7 void auth_withpeer_success (int, u16_t)

5.95.1.8 int bad_ip_adrs (u32_t)

5.95.1.9 u_char check_passwd (int, char *, int, char *, int, char **, int *)

5.95.1.10 int get_secret (int, char *, char *, char *, int *, int)

5.95.1.11 void link_down (int)

5.95.1.12 void link_established (int)

5.95.1.13 void link_required (int)

5.95.1.14 void link_terminated (int)

5.95.1.15 void np_down (int, u16_t)

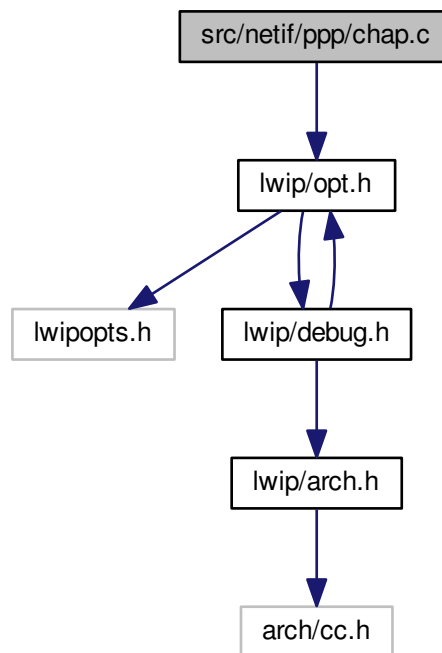
5.95.1.16 void np_finished (int, u16_t)

5.95.1.17 void np_up (int, u16_t)

5.96 src/netif/ppp/chap.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for chap.c:



5.97 src/netif/ppp/chap.h File Reference

Data Structures

- struct [chap_state](#)

Macros

- #define [CHAP_HEADERLEN](#) 4
- #define [CHAP_DIGEST_MD5](#) 5 /* use MD5 algorithm */
- #define [MD5_SIGNATURE_SIZE](#) 16 /* 16 bytes in a MD5 message digest */
- #define [CHAP_MICROSOFT](#) 0x80 /* use Microsoft-compatible alg. */
- #define [MS_CHAP_RESPONSE_LEN](#) 49 /* Response length for MS-CHAP */
- #define [CHAP_CHALLENGE](#) 1
- #define [CHAP_RESPONSE](#) 2
- #define [CHAP_SUCCESS](#) 3
- #define [CHAP_FAILURE](#) 4
- #define [MIN_CHALLENGE_LENGTH](#) 32
- #define [MAX_CHALLENGE_LENGTH](#) 64
- #define [MAX_RESPONSE_LENGTH](#) 64 /* sufficient for MD5 or MS-CHAP */
- #define [CHAPCS_INITIAL](#) 0 /* Lower layer down, not opened */
- #define [CHAPCS_CLOSED](#) 1 /* Lower layer up, not opened */
- #define [CHAPCS_PENDING](#) 2 /* Auth us to peer when lower up */
- #define [CHAPCS_LISTEN](#) 3 /* Listening for a challenge */

- `#define CHAPCS_RESPONSE 4` /* Sent response, waiting for status */
- `#define CHAPCS_OPEN 5` /* We've received Success */
- `#define CHAPSS_INITIAL 0` /* Lower layer down, not opened */
- `#define CHAPSS_CLOSED 1` /* Lower layer up, not opened */
- `#define CHAPSS_PENDING 2` /* Auth peer when lower up */
- `#define CHAPSS_INITIAL_CHAL 3` /* We've sent the first challenge */
- `#define CHAPSS_OPEN 4` /* We've sent a Success msg */
- `#define CHAPSS_RECHALLENGE 5` /* We've sent another challenge */
- `#define CHAPSS_BADAUTH 6` /* We've sent a Failure msg */

Typedefs

- typedef struct `chap_state` `chap_state`

Functions

- void `ChapAuthWithPeer` (int, char *, u_char)
- void `ChapAuthPeer` (int, char *, u_char)

Variables

- `chap_state` `chap` []
- struct protent `chap_protent`

5.97.1 Macro Definition Documentation

5.97.1.1 `#define CHAP_CHALLENGE 1`

Definition at line 83 of file chap.h.

5.97.1.2 `#define CHAP_DIGEST_MD5 5` /* use MD5 algorithm */

Definition at line 78 of file chap.h.

5.97.1.3 `#define CHAP_FAILURE 4`

Definition at line 86 of file chap.h.

5.97.1.4 `#define CHAP_HEADERLEN 4`

Definition at line 72 of file chap.h.

5.97.1.5 `#define CHAP_MICROSOFT 0x80` /* use Microsoft-compatible alg. */

Definition at line 80 of file chap.h.

5.97.1.6 `#define CHAP_RESPONSE 2`

Definition at line 84 of file chap.h.

5.97.1.7 #define CHAP_SUCCESS 3

Definition at line 85 of file chap.h.

5.97.1.8 #define CHAPCS_CLOSED 1 /* Lower layer up, not opened */

Definition at line 126 of file chap.h.

5.97.1.9 #define CHAPCS_INITIAL 0 /* Lower layer down, not opened */

Definition at line 125 of file chap.h.

5.97.1.10 #define CHAPCS_LISTEN 3 /* Listening for a challenge */

Definition at line 128 of file chap.h.

5.97.1.11 #define CHAPCS_OPEN 5 /* We've received Success */

Definition at line 130 of file chap.h.

5.97.1.12 #define CHAPCS_PENDING 2 /* Auth us to peer when lower up */

Definition at line 127 of file chap.h.

5.97.1.13 #define CHAPCS_RESPONSE 4 /* Sent response, waiting for status */

Definition at line 129 of file chap.h.

5.97.1.14 #define CHAPSS_BDAUTH 6 /* We've sent a Failure msg */

Definition at line 141 of file chap.h.

5.97.1.15 #define CHAPSS_CLOSED 1 /* Lower layer up, not opened */

Definition at line 136 of file chap.h.

5.97.1.16 #define CHAPSS_INITIAL 0 /* Lower layer down, not opened */

Definition at line 135 of file chap.h.

5.97.1.17 #define CHAPSS_INITIAL_CHAL 3 /* We've sent the first challenge */

Definition at line 138 of file chap.h.

5.97.1.18 #define CHAPSS_OPEN 4 /* We've sent a Success msg */

Definition at line 139 of file chap.h.

5.97.1.19 `#define CHAPSS_PENDING 2 /* Auth peer when lower up */`

Definition at line 137 of file chap.h.

5.97.1.20 `#define CHAPSS_RECHALLENGE 5 /* We've sent another challenge */`

Definition at line 140 of file chap.h.

5.97.1.21 `#define MAX_CHALLENGE_LENGTH 64`

Definition at line 92 of file chap.h.

5.97.1.22 `#define MAX_RESPONSE_LENGTH 64 /* sufficient for MD5 or MS-CHAP */`

Definition at line 93 of file chap.h.

5.97.1.23 `#define MD5_SIGNATURE_SIZE 16 /* 16 bytes in a MD5 message digest */`

Definition at line 79 of file chap.h.

5.97.1.24 `#define MIN_CHALLENGE_LENGTH 32`

Definition at line 91 of file chap.h.

5.97.1.25 `#define MS_CHAP_RESPONSE_LEN 49 /* Response length for MS-CHAP */`

Definition at line 81 of file chap.h.

5.97.2 Typedef Documentation

5.97.2.1 `typedef struct chap_state chap_state`

5.97.3 Function Documentation

5.97.3.1 `void ChapAuthPeer (int , char * , u_char)`

5.97.3.2 `void ChapAuthWithPeer (int , char * , u_char)`

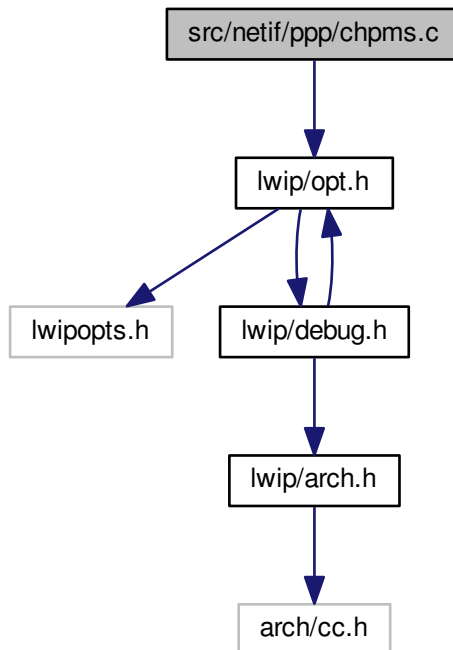
5.97.4 Variable Documentation

5.97.4.1 `chap_state chap[]`

5.97.4.2 `struct protent chap_protent`

5.98 src/netif/ppp/chpms.c File Reference

```
#include "lwip/opt.h"
Include dependency graph for chpms.c:
```



Macros

- `#define` [USE_CRYPT](#)

5.98.1 Macro Definition Documentation

5.98.1.1 `#define` [USE_CRYPT](#)

Definition at line 70 of file `chpms.c`.

5.99 src/netif/ppp/chpms.h File Reference

Macros

- `#define` [MAX_NT_PASSWORD](#) 256 /* Maximum number of (Unicode) chars in an NT password */

Functions

- void [ChapMS](#) ([chap_state](#) *, char *, int, char *, int)

5.99.1 Macro Definition Documentation

5.99.1.1 `#define MAX_NT_PASSWORD 256 /* Maximum number of (Unicode) chars in an NT password */`

Definition at line 60 of file `chpms.h`.

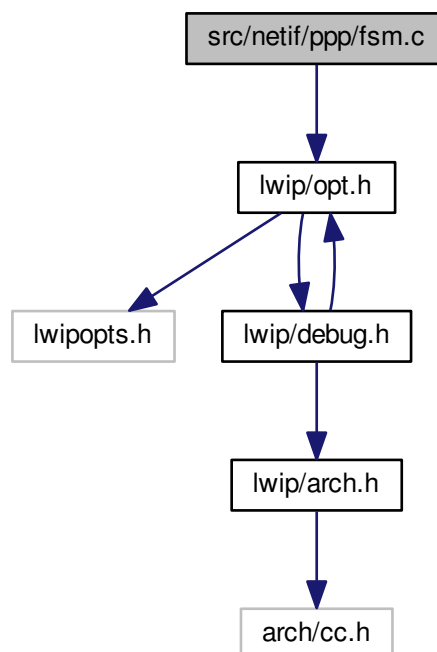
5.99.2 Function Documentation

5.99.2.1 `void ChapMS (chap_state *, char *, int, char *, int)`

5.100 `src/netif/ppp/fsm.c` File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for `fsm.c`:



5.101 `src/netif/ppp/fsm.h` File Reference

Data Structures

- struct [fsm](#)
- struct [fsm_callbacks](#)

Macros

- `#define HEADERLEN (sizeof (u_char) + sizeof (u_char) + sizeof (u_short))`

- `#define CONFREQ 1 /* Configuration Request */`
- `#define CONFACK 2 /* Configuration Ack */`
- `#define CONFNAK 3 /* Configuration Nak */`
- `#define CONFREJ 4 /* Configuration Reject */`
- `#define TERMREQ 5 /* Termination Request */`
- `#define TERMACK 6 /* Termination Ack */`
- `#define CODEREJ 7 /* Code Reject */`
- `#define LS_INITIAL 0 /* Down, hasn't been opened */`
- `#define LS_STARTING 1 /* Down, been opened */`
- `#define LS_CLOSED 2 /* Up, hasn't been opened */`
- `#define LS_STOPPED 3 /* Open, waiting for down event */`
- `#define LS_CLOSING 4 /* Terminating the connection, not open */`
- `#define LS_STOPPING 5 /* Terminating, but open */`
- `#define LS_REQSENT 6 /* We've sent a Config Request */`
- `#define LS_ACKRCVD 7 /* We've received a Config Ack */`
- `#define LS_ACKSENT 8 /* We've sent a Config Ack */`
- `#define LS_OPENED 9 /* Connection available */`
- `#define OPT_PASSIVE 1 /* Don't die if we don't get a response */`
- `#define OPT_RESTART 2 /* Treat 2nd OPEN as DOWN, UP */`
- `#define OPT_SILENT 4 /* Wait for peer to speak first */`

Typedefs

- `typedef struct fsm fsm`
- `typedef struct fsm_callbacks fsm_callbacks`

Functions

- `void fsm_init (fsm *)`
- `void fsm_lowerup (fsm *)`
- `void fsm_lowerdown (fsm *)`
- `void fsm_open (fsm *)`
- `void fsm_close (fsm *, char *)`
- `void fsm_input (fsm *, u_char *, int)`
- `void fsm_protreject (fsm *)`
- `void fsm_sdata (fsm *, u_char, u_char, u_char *, int)`

Variables

- `int peer_mru []`

5.101.1 Macro Definition Documentation

5.101.1.1 `#define CODEREJ 7 /* Code Reject */`

Definition at line 72 of file fsm.h.

5.101.1.2 `#define CONFACK 2 /* Configuration Ack */`

Definition at line 67 of file fsm.h.

5.101.1.3 **#define CONFNAK 3 /* Configuration Nak */**

Definition at line 68 of file fsm.h.

5.101.1.4 **#define CONFREJ 4 /* Configuration Reject */**

Definition at line 69 of file fsm.h.

5.101.1.5 **#define CONFREQ 1 /* Configuration Request */**

Definition at line 66 of file fsm.h.

5.101.1.6 **#define HEADERLEN (sizeof (u_char) + sizeof (u_char) + sizeof (u_short))**

Definition at line 60 of file fsm.h.

5.101.1.7 **#define LS_ACKRCVD 7 /* We've received a Config Ack */**

Definition at line 127 of file fsm.h.

5.101.1.8 **#define LS_ACKSENT 8 /* We've sent a Config Ack */**

Definition at line 128 of file fsm.h.

5.101.1.9 **#define LS_CLOSED 2 /* Up, hasn't been opened */**

Definition at line 122 of file fsm.h.

5.101.1.10 **#define LS_CLOSING 4 /* Terminating the connection, not open */**

Definition at line 124 of file fsm.h.

5.101.1.11 **#define LS_INITIAL 0 /* Down, hasn't been opened */**

Definition at line 120 of file fsm.h.

5.101.1.12 **#define LS_OPENED 9 /* Connection available */**

Definition at line 129 of file fsm.h.

5.101.1.13 **#define LS_REQSENT 6 /* We've sent a Config Request */**

Definition at line 126 of file fsm.h.

5.101.1.14 **#define LS_STARTING 1 /* Down, been opened */**

Definition at line 121 of file fsm.h.

5.101.1.15 `#define LS_STOPPED 3 /* Open, waiting for down event */`

Definition at line 123 of file fsm.h.

5.101.1.16 `#define LS_STOPPING 5 /* Terminating, but open */`

Definition at line 125 of file fsm.h.

5.101.1.17 `#define OPT_PASSIVE 1 /* Don't die if we don't get a response */`

Definition at line 134 of file fsm.h.

5.101.1.18 `#define OPT_RESTART 2 /* Treat 2nd OPEN as DOWN, UP */`

Definition at line 135 of file fsm.h.

5.101.1.19 `#define OPT_SILENT 4 /* Wait for peer to speak first */`

Definition at line 136 of file fsm.h.

5.101.1.20 `#define TERMACK 6 /* Termination Ack */`

Definition at line 71 of file fsm.h.

5.101.1.21 `#define TERMREQ 5 /* Termination Request */`

Definition at line 70 of file fsm.h.

5.101.2 Typedef Documentation

5.101.2.1 `typedef struct fsm fsm`

5.101.2.2 `typedef struct fsm_callbacks fsm_callbacks`

5.101.3 Function Documentation

5.101.3.1 `void fsm_close (fsm *, char *)`

5.101.3.2 `void fsm_init (fsm *)`

5.101.3.3 `void fsm_input (fsm *, u_char *, int)`

5.101.3.4 `void fsm_lowerdown (fsm *)`

5.101.3.5 `void fsm_lowerup (fsm *)`

5.101.3.6 `void fsm_open (fsm *)`

5.101.3.7 `void fsm_protreject (fsm *)`

5.101.3.8 `void fsm_sdata (fsm *, u_char, u_char, u_char *, int)`

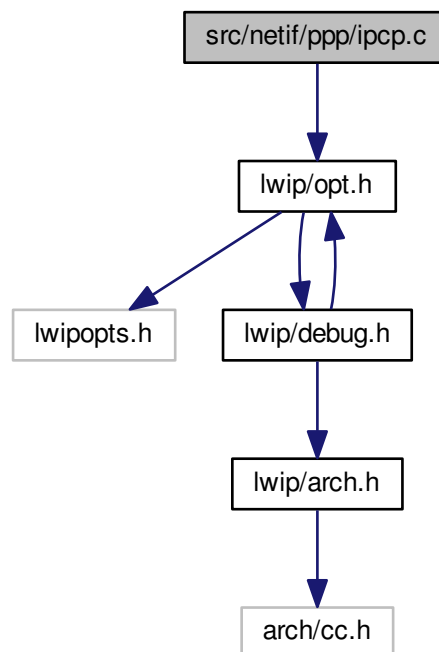
5.101.4 Variable Documentation

5.101.4.1 `int peer_mru[]`

5.102 `src/netif/ppp/ipcp.c` File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for `ipcp.c`:



5.103 `src/netif/ppp/ipcp.h` File Reference

Data Structures

- struct [ipcp_options](#)

Macros

- `#define CI_ADDRS 1` /* IP Addresses */
- `#define CI_COMPRESSTYPE 2` /* Compression Type */
- `#define CI_ADDR 3`
- `#define CI_MS_DNS1 129` /* Primary DNS value */
- `#define CI_MS_WINS1 128` /* Primary WINS value */
- `#define CI_MS_DNS2 131` /* Secondary DNS value */
- `#define CI_MS_WINS2 130` /* Secondary WINS value */
- `#define IPCP_VJMODE_OLD 1` /* "old" mode (option # = 0x0037) */

- `#define IPCP_VJMODE_RFC1172 2` /* "old-rfc" mode (option # = 0x002d) */
- `#define IPCP_VJMODE_RFC1332 3` /* "new-rfc" mode (option # = 0x002d, */
- `#define IPCP_VJ_COMP 0x002d` /* current value for VJ compression option */
- `#define IPCP_VJ_COMP_OLD 0x0037` /* "old" (i.e, broken) value for VJ */

Typedefs

- typedef struct `ipcp_options` `ipcp_options`

Variables

- fsm `ipcp_fsm` []
- `ipcp_options` `ipcp_wantoptions` []
- `ipcp_options` `ipcp_gotoptions` []
- `ipcp_options` `ipcp_allowoptions` []
- `ipcp_options` `ipcp_hisoptions` []
- struct protent `ipcp_protent`

5.103.1 Macro Definition Documentation

5.103.1.1 `#define CI_ADDR 3`

Definition at line 62 of file `ipcp.h`.

5.103.1.2 `#define CI_ADDRS 1` /* IP Addresses */

Definition at line 60 of file `ipcp.h`.

5.103.1.3 `#define CI_COMPRESSTYPE 2` /* Compression Type */

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

5.103.1.4 `#define CI_MS_DNS1 129` /* Primary DNS value */

Definition at line 64 of file `ipcp.h`.

5.103.1.5 `#define CI_MS_DNS2 131` /* Secondary DNS value */

Definition at line 66 of file `ipcp.h`.

5.103.1.6 `#define CI_MS_WINS1 128` /* Primary WINS value */

Definition at line 65 of file `ipcp.h`.

5.103.1.7 `#define CI_MS_WINS2 130` /* Secondary WINS value */

Definition at line 67 of file `ipcp.h`.

5.103.1.8 `#define IPCP_VJ_COMP 0x002d /* current value for VJ compression option */`

Definition at line 74 of file `ipcp.h`.

5.103.1.9 `#define IPCP_VJ_COMP_OLD 0x0037 /* "old" (i.e, broken) value for VJ */`

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

5.103.1.10 `#define IPCP_VJMODE_OLD 1 /* "old" mode (option # = 0x0037) */`

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

5.103.1.11 `#define IPCP_VJMODE_RFC1172 2 /* "old-rfc" mode (option # = 0x002d) */`

Definition at line 70 of file `ipcp.h`.

5.103.1.12 `#define IPCP_VJMODE_RFC1332 3 /* "new-rfc" mode (option # = 0x002d, */`

Definition at line 71 of file `ipcp.h`.

5.103.2 Typedef Documentation

5.103.2.1 `typedef struct ipcp_options ipcp_options`

5.103.3 Variable Documentation

5.103.3.1 `ipcp_options ipcp_allowoptions[]`

5.103.3.2 `fsm ipcp_fsm[]`

5.103.3.3 `ipcp_options ipcp_gotoptions[]`

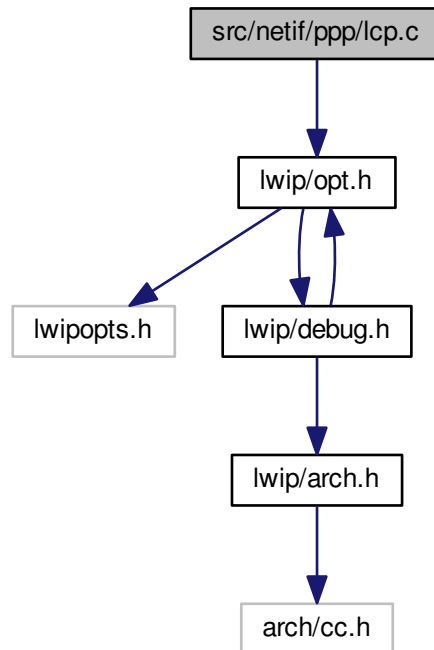
5.103.3.4 `ipcp_options ipcp_hisoptions[]`

5.103.3.5 `struct protent ipcp_protent`

5.103.3.6 `ipcp_options ipcp_wantoptions[]`

5.104 src/netif/ppp/lcp.c File Reference

```
#include "lwip/opt.h"
Include dependency graph for lcp.c:
```



5.105 src/netif/ppp/lcp.h File Reference

Data Structures

- struct [lcp_options](#)

Macros

- `#define` [CI_MRU](#) 1 /* Maximum Receive Unit */
- `#define` [CI_ASYNCMAP](#) 2 /* Async Control Character Map */
- `#define` [CI_AUTHTYPE](#) 3 /* Authentication Type */
- `#define` [CI_QUALITY](#) 4 /* Quality Protocol */
- `#define` [CI_MAGICNUMBER](#) 5 /* Magic Number */
- `#define` [CI_PCOMPRESSION](#) 7 /* Protocol Field Compression */
- `#define` [CI_ACCOMPRESSION](#) 8 /* Address/Control Field Compression */
- `#define` [CI_CALLBACK](#) 13 /* callback */
- `#define` [CI_MRRU](#) 17 /* max reconstructed receive unit; multilink */
- `#define` [CI_SSNHF](#) 18 /* short sequence numbers for multilink */
- `#define` [CI_EPDISC](#) 19 /* endpoint discriminator */
- `#define` [PROTREJ](#) 8 /* Protocol Reject */

- `#define ECHOREQ 9 /* Echo Request */`
- `#define ECHOREP 10 /* Echo Reply */`
- `#define DISCREQ 11 /* Discard Request */`
- `#define CBCP_OPT 6 /* Use callback control protocol */`
- `#define DEFLOOPBACKFAIL 10`

Typedefs

- typedef struct `lcp_options` `lcp_options`

Enumerations

- enum `LinkPhase` {
`PHASE_DEAD = 0, PHASE_INITIALIZE, PHASE_ESTABLISH, PHASE_AUTHENTICATE,`
`PHASE_CALLBACK, PHASE_NETWORK, PHASE_TERMINATE` }

Functions

- void `lcp_init` (int)
- void `lcp_open` (int)
- void `lcp_close` (int, char *)
- void `lcp_lowerup` (int)
- void `lcp_lowerdown` (int)
- void `lcp_sprotrej` (int, u_char *, int)

Variables

- `LinkPhase` `lcp_phase` [NUM_PPP]
- `lcp_options` `lcp_wantoptions` []
- `lcp_options` `lcp_gotoptions` []
- `lcp_options` `lcp_allowoptions` []
- `lcp_options` `lcp_hisoptions` []
- ext_accm `xmit_accm` []
- struct protent `lcp_protent`

5.105.1 Macro Definition Documentation

5.105.1.1 `#define CBCP_OPT 6 /* Use callback control protocol */`

Definition at line 78 of file `lcp.h`.

5.105.1.2 `#define CI_ACCOMPRESSSION 8 /* Address/Control Field Compression */`

Definition at line 65 of file `lcp.h`.

5.105.1.3 `#define CI_ASYNCMAP 2 /* Async Control Character Map */`

Definition at line 60 of file `lcp.h`.

5.105.1.4 #define CI_AUTHTYPE 3 /* Authentication Type */

Definition at line 61 of file lcp.h.

5.105.1.5 #define CI_CALLBACK 13 /* callback */

Definition at line 66 of file lcp.h.

5.105.1.6 #define CI_EPDISC 19 /* endpoint discriminator */

Definition at line 69 of file lcp.h.

5.105.1.7 #define CI_MAGICNUMBER 5 /* Magic Number */

Definition at line 63 of file lcp.h.

5.105.1.8 #define CI_MRRU 17 /* max reconstructed receive unit; multilink */

Definition at line 67 of file lcp.h.

5.105.1.9 #define CI_MRU 1 /* Maximum Receive Unit */

Definition at line 59 of file lcp.h.

5.105.1.10 #define CI_PCOMPRESSION 7 /* Protocol Field Compression */

Definition at line 64 of file lcp.h.

5.105.1.11 #define CI_QUALITY 4 /* Quality Protocol */

Definition at line 62 of file lcp.h.

5.105.1.12 #define CI_SSNHF 18 /* short sequence numbers for multilink */

Definition at line 68 of file lcp.h.

5.105.1.13 #define DEFLOOPBACKFAIL 10

Definition at line 149 of file lcp.h.

5.105.1.14 #define DISCREQ 11 /* Discard Request */

Definition at line 77 of file lcp.h.

5.105.1.15 #define ECHOREP 10 /* Echo Reply */

Definition at line 76 of file lcp.h.

5.105.1.16 `#define ECHOREQ 9 /* Echo Request */`

Definition at line 75 of file lcp.h.

5.105.1.17 `#define PROTREJ 8 /* Protocol Reject */`

Definition at line 74 of file lcp.h.

5.105.2 Typedef Documentation

5.105.2.1 `typedef struct lcp_options lcp_options`

5.105.3 Enumeration Type Documentation

5.105.3.1 `enum LinkPhase`

Enumerator

PHASE_DEAD

PHASE_INITIALIZE

PHASE_ESTABLISH

PHASE_AUTHENTICATE

PHASE_CALLBACK

PHASE_NETWORK

PHASE_TERMINATE

Definition at line 118 of file lcp.h.

5.105.4 Function Documentation

5.105.4.1 `void lcp_close (int , char *)`

5.105.4.2 `void lcp_init (int)`

5.105.4.3 `void lcp_lowerdown (int)`

5.105.4.4 `void lcp_lowerup (int)`

5.105.4.5 `void lcp_open (int)`

5.105.4.6 `void lcp_sprotrej (int , u_char * , int)`

5.105.5 Variable Documentation

5.105.5.1 `lcp_options lcp_allowoptions[]`

5.105.5.2 `lcp_options lcp_gotoptions[]`

5.105.5.3 `lcp_options lcp_hisoptions[]`

5.105.5.4 `LinkPhase lcp_phase[NUM_PPP]`

5.105.5.5 `struct protent lcp_protent`

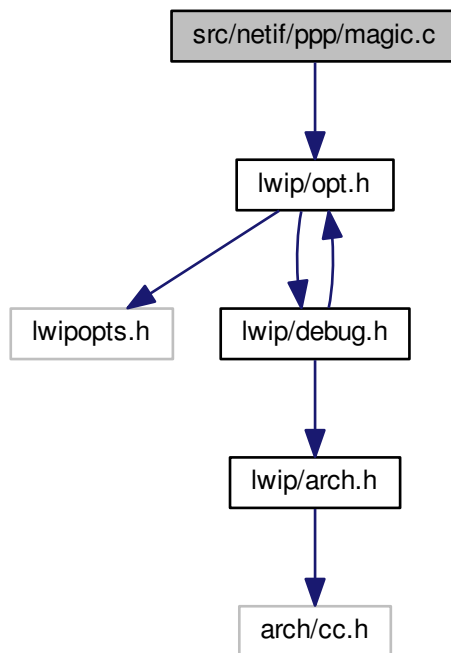
5.105.5.6 lcp_options lcp_wantoptions[]

5.105.5.7 ext_accm xmit_accm[]

5.106 src/netif/ppp/magic.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for magic.c:



5.107 src/netif/ppp/magic.h File Reference

Functions

- void [magicInit](#) (void)
- u32_t [magic](#) (void)

5.107.1 Function Documentation

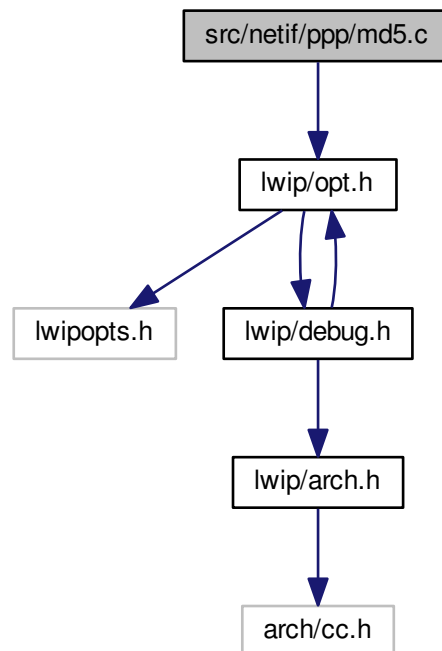
5.107.1.1 u32_t [magic](#) (void)

5.107.1.2 void [magicInit](#) (void)

5.108 src/netif/ppp/md5.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for md5.c:



5.109 src/netif/ppp/md5.h File Reference

Data Structures

- struct [MD5_CTX](#)

Functions

- void [MD5Init](#) ([MD5_CTX](#) *mdContext)
- void [MD5Update](#) ([MD5_CTX](#) *mdContext, unsigned char *inBuf, unsigned int inLen)
- void [MD5Final](#) (unsigned char hash[], [MD5_CTX](#) *mdContext)

5.109.1 Function Documentation

5.109.1.1 void [MD5Final](#) (unsigned char *hash*[], [MD5_CTX](#) * *mdContext*)

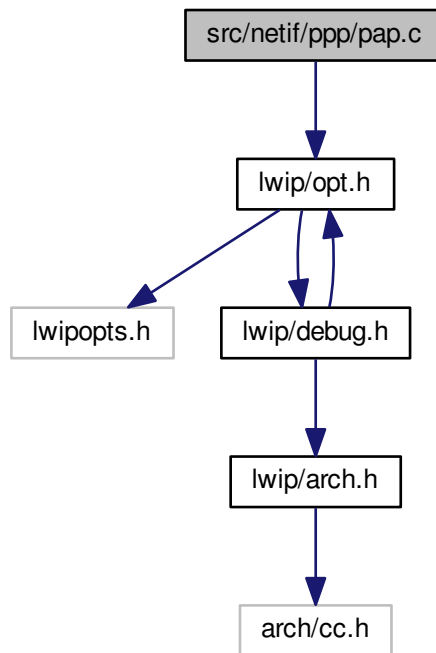
5.109.1.2 void [MD5Init](#) ([MD5_CTX](#) * *mdContext*)

5.109.1.3 void [MD5Update](#) ([MD5_CTX](#) * *mdContext*, unsigned char * *inBuf*, unsigned int *inLen*)

5.110 src/netif/ppp/pap.c File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for pap.c:

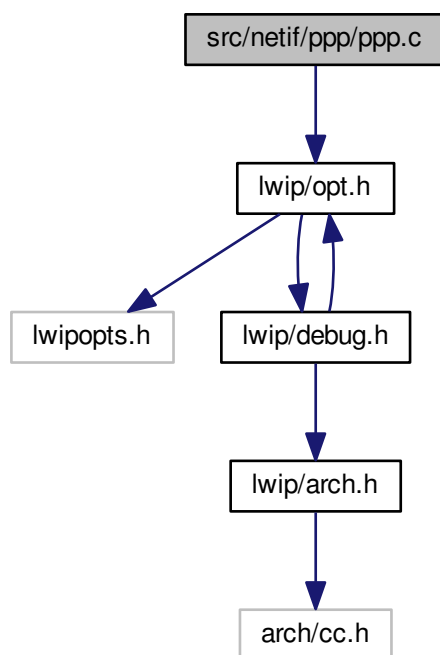


5.111 src/netif/ppp/pap.h File Reference

5.112 src/netif/ppp/ppp.c File Reference

```
#include "lwip/opt.h"
```

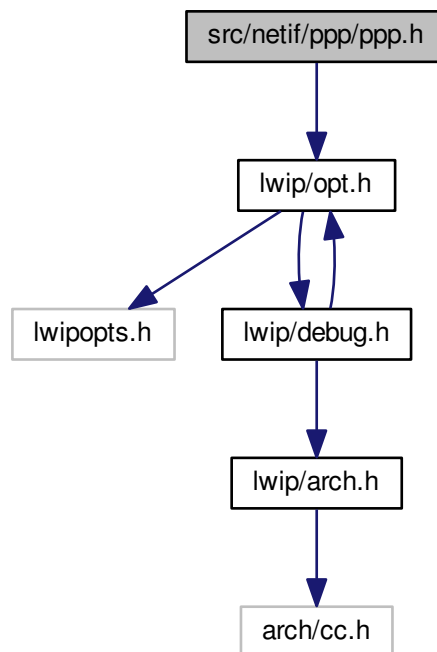
Include dependency graph for ppp.c:



5.113 src/netif/ppp/ppp.h File Reference

```
#include "lwip/opt.h"
```

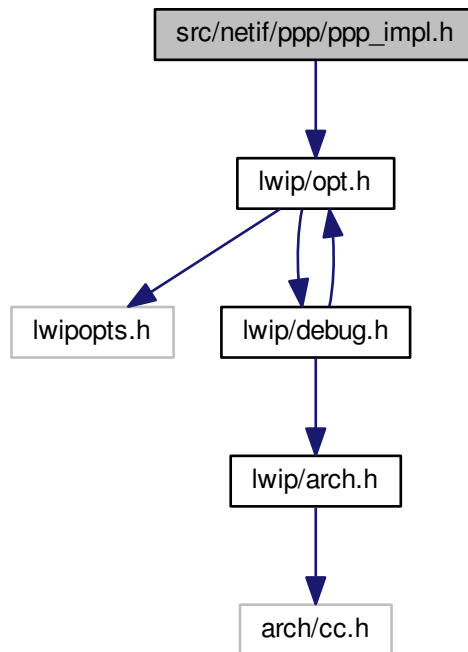
Include dependency graph for ppp.h:



5.114 src/netif/ppp/ppp_impl.h File Reference

```
#include "lwip/opt.h"
```

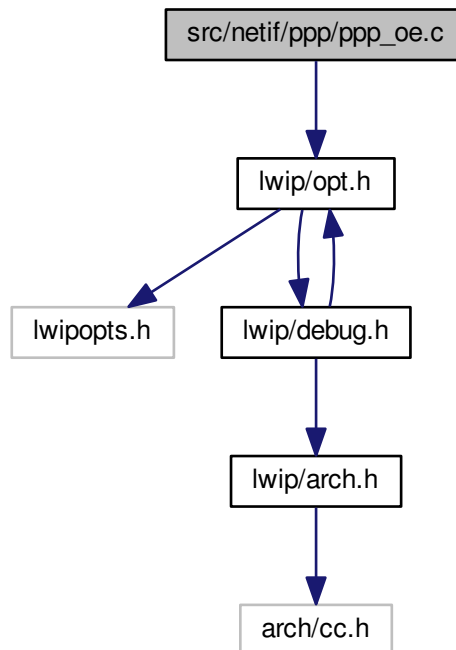
Include dependency graph for ppp_impl.h:



5.115 src/netif/ppp/ppp_oe.c File Reference

```
#include "lwip/opt.h"
```


Include dependency graph for ppp_oe.c:



5.116 src/netif/ppp/pppdebug.h File Reference

Macros

- `#define LOG_CRITICAL (PPP_DEBUG | LWIP_DBG_LEVEL_SEVERE)`
- `#define LOG_ERR (PPP_DEBUG | LWIP_DBG_LEVEL_SEVERE)`
- `#define LOG_NOTICE (PPP_DEBUG | LWIP_DBG_LEVEL_WARNING)`
- `#define LOG_WARNING (PPP_DEBUG | LWIP_DBG_LEVEL_WARNING)`
- `#define LOG_INFO (PPP_DEBUG)`
- `#define LOG_DETAIL (PPP_DEBUG)`
- `#define LOG_DEBUG (PPP_DEBUG)`
- `#define TRACELCP PPP_DEBUG`
- `#define AUTHDEBUG(a, b)`
- `#define IPCPDEBUG(a, b)`
- `#define UPAPDEBUG(a, b)`
- `#define LCPDEBUG(a, b)`
- `#define FSMDEBUG(a, b)`
- `#define CHAPDEBUG(a, b)`
- `#define PPPDEBUG(a, b)`

5.116.1 Macro Definition Documentation

5.116.1.1 `#define AUTHDEBUG(a, b)`

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

5.116.1.2 **#define CHAPDEBUG(a, b)**

Definition at line 68 of file pppdebug.h.

5.116.1.3 **#define FSMDEBUG(a, b)**

Definition at line 67 of file pppdebug.h.

5.116.1.4 **#define IPCPDEBUG(a, b)**

Definition at line 64 of file pppdebug.h.

5.116.1.5 **#define LCPDEBUG(a, b)**

Definition at line 66 of file pppdebug.h.

5.116.1.6 **#define LOG_CRITICAL (PPP_DEBUG | LWIP_DBG_LEVEL_SEVERE)**

Definition at line 40 of file pppdebug.h.

5.116.1.7 **#define LOG_DEBUG (PPP_DEBUG)**

Definition at line 46 of file pppdebug.h.

5.116.1.8 **#define LOG_DETAIL (PPP_DEBUG)**

Definition at line 45 of file pppdebug.h.

5.116.1.9 **#define LOG_ERR (PPP_DEBUG | LWIP_DBG_LEVEL_SEVERE)**

Definition at line 41 of file pppdebug.h.

5.116.1.10 **#define LOG_INFO (PPP_DEBUG)**

Definition at line 44 of file pppdebug.h.

5.116.1.11 **#define LOG_NOTICE (PPP_DEBUG | LWIP_DBG_LEVEL_WARNING)**

Definition at line 42 of file pppdebug.h.

5.116.1.12 **#define LOG_WARNING (PPP_DEBUG | LWIP_DBG_LEVEL_WARNING)**

Definition at line 43 of file pppdebug.h.

5.116.1.13 **#define PPPDEBUG(a, b)**

Definition at line 69 of file pppdebug.h.

5.116.1.14 `#define TRACE_LCP PPP_DEBUG`

Definition at line 49 of file pppdebug.h.

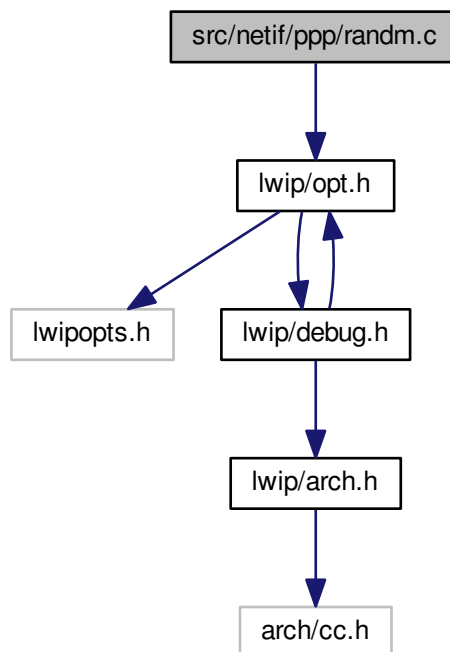
5.116.1.15 `#define UPAPDEBUG(a, b)`

Definition at line 65 of file pppdebug.h.

5.117 `src/netif/ppp/randm.c` File Reference

```
#include "lwip/opt.h"
```

Include dependency graph for randm.c:

5.118 `src/netif/ppp/randm.h` File Reference

Functions

- void [avRandomInit](#) (void)
- void [avChurnRand](#) (char *randData, u32_t randLen)
- void [avRandomize](#) (void)
- void [avGenRand](#) (char *buf, u32_t bufLen)
- u32_t [avRandom](#) (void)

5.118.1 Function Documentation

5.118.1.1 void avChurnRand (char * *randData*, u32_t *randLen*)

5.118.1.2 void avGenRand (char * *buf*, u32_t *bufLen*)

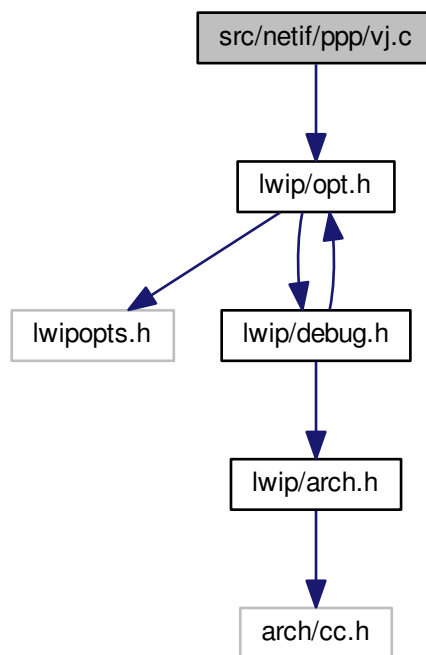
5.118.1.3 u32_t avRandom (void)

5.118.1.4 void avRandomInit (void)

5.118.1.5 void avRandomize (void)

5.119 src/netif/ppp/vj.c File Reference

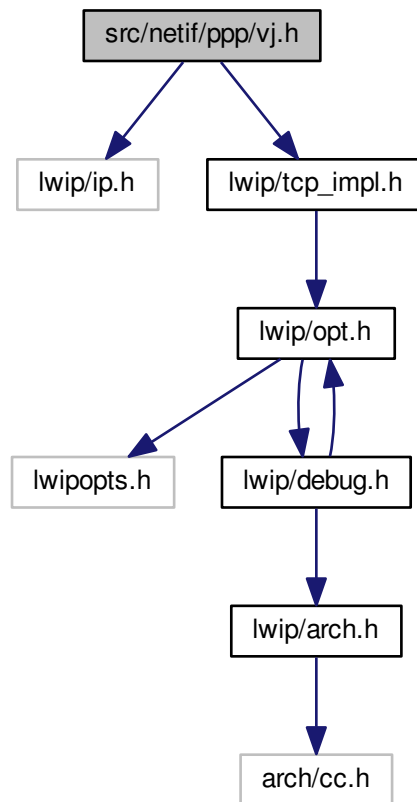
```
#include "lwip/opt.h"
Include dependency graph for vj.c:
```



5.120 src/netif/ppp/vj.h File Reference

```
#include "lwip/ip.h"
#include "lwip/tcp_impl.h"
```

Include dependency graph for vj.h:



Data Structures

- struct [cstate](#)
- struct [vjstat](#)
- struct [vjcompress](#)

Macros

- #define [MAX_SLOTS](#) 16 /* must be > 2 and < 256 */
- #define [MAX_HDR](#) 128
- #define [TYPE_IP](#) 0x40
- #define [TYPE_UNCOMPRESSED_TCP](#) 0x70
- #define [TYPE_COMPRESSED_TCP](#) 0x80
- #define [TYPE_ERROR](#) 0x00
- #define [NEW_C](#) 0x40 /* flag bits for what changed in a packet */
- #define [NEW_I](#) 0x20
- #define [NEW_S](#) 0x08
- #define [NEW_A](#) 0x04
- #define [NEW_W](#) 0x02
- #define [NEW_U](#) 0x01

- `#define SPECIAL_I (NEW_S|NEW_W|NEW_U) /* echoed interactive traffic */`
- `#define SPECIAL_D (NEW_S|NEW_A|NEW_W|NEW_U) /* unidirectional data */`
- `#define SPECIALS_MASK (NEW_S|NEW_A|NEW_W|NEW_U)`
- `#define TCP_PUSH_BIT 0x10`
- `#define cs_ip vjcs_u.csu_ip`
- `#define cs_hdr vjcs_u.csu_hdr`
- `#define VJF_TOSS 1U /* tossing rcvd frames because of input err */`

Functions

- void `vj_compress_init` (struct `vjcompress` *comp)
- u_int `vj_compress_tcp` (struct `vjcompress` *comp, struct `pbuf` *pb)
- void `vj_uncompress_err` (struct `vjcompress` *comp)
- int `vj_uncompress_uncomp` (struct `pbuf` *nb, struct `vjcompress` *comp)
- int `vj_uncompress_tcp` (struct `pbuf` **nb, struct `vjcompress` *comp)

5.120.1 Macro Definition Documentation

5.120.1.1 `#define cs_hdr vjcs_u.csu_hdr`

Definition at line 116 of file `vj.h`.

5.120.1.2 `#define cs_ip vjcs_u.csu_ip`

Definition at line 115 of file `vj.h`.

5.120.1.3 `#define MAX_HDR 128`

Definition at line 32 of file `vj.h`.

5.120.1.4 `#define MAX_SLOTS 16 /* must be > 2 and < 256 */`

Definition at line 31 of file `vj.h`.

5.120.1.5 `#define NEW_A 0x04`

Definition at line 87 of file `vj.h`.

5.120.1.6 `#define NEW_C 0x40 /* flag bits for what changed in a packet */`

Definition at line 84 of file `vj.h`.

5.120.1.7 `#define NEW_I 0x20`

Definition at line 85 of file `vj.h`.

5.120.1.8 `#define NEW_S 0x08`

Definition at line 86 of file `vj.h`.

5.120.1.9 `#define NEW_U 0x01`

Definition at line 89 of file vj.h.

5.120.1.10 `#define NEW_W 0x02`

Definition at line 88 of file vj.h.

5.120.1.11 `#define SPECIAL_D (NEW_S|NEW_A|NEW_W|NEW_U) /* unidirectional data */`

Definition at line 93 of file vj.h.

5.120.1.12 `#define SPECIAL_I (NEW_S|NEW_W|NEW_U) /* echoed interactive traffic */`

Definition at line 92 of file vj.h.

5.120.1.13 `#define SPECIALS_MASK (NEW_S|NEW_A|NEW_W|NEW_U)`

Definition at line 94 of file vj.h.

5.120.1.14 `#define TCP_PUSH_BIT 0x10`

Definition at line 96 of file vj.h.

5.120.1.15 `#define TYPE_COMPRESSED_TCP 0x80`

Definition at line 80 of file vj.h.

5.120.1.16 `#define TYPE_ERROR 0x00`

Definition at line 81 of file vj.h.

5.120.1.17 `#define TYPE_IP 0x40`

Definition at line 78 of file vj.h.

5.120.1.18 `#define TYPE_UNCOMPRESSED_TCP 0x70`

Definition at line 79 of file vj.h.

5.120.1.19 `#define VJF_TOSS 1U /* tossing rcvd frames because of input err */`

Definition at line 148 of file vj.h.

5.120.2 Function Documentation

5.120.2.1 `void vj_compress_init (struct vjcompress * comp)`

5.120.2.2 `u_int vj_compress_tcp (struct vjcompress * comp, struct pbuf * pb)`

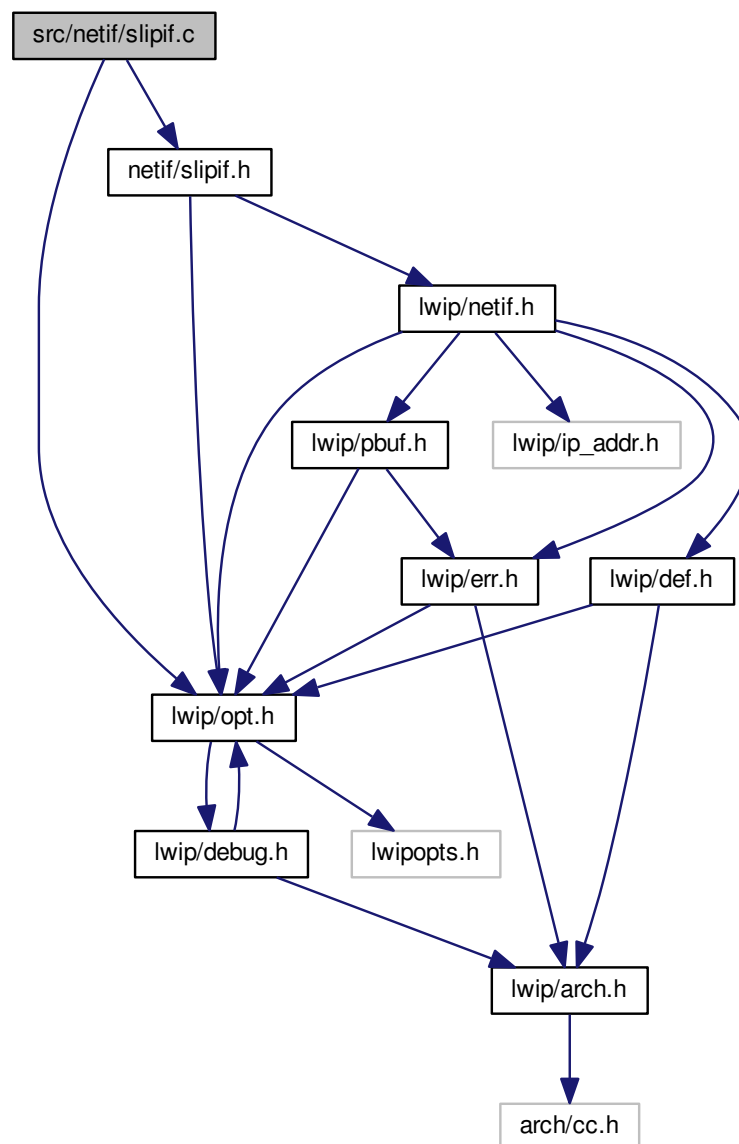
5.120.2.3 void vj_uncompress_err (struct vjcompress * comp)

5.120.2.4 int vj_uncompress_tcp (struct pbuf ** nb, struct vjcompress * comp)

5.120.2.5 int vj_uncompress_uncomp (struct pbuf * nb, struct vjcompress * comp)

5.121 src/netif/slipif.c File Reference

```
#include "netif/slipif.h"
#include "lwip/opt.h"
Include dependency graph for slipif.c:
```



5.121.1 Detailed Description

SLIP Interface

Index

- API_LIB_DEBUG
 - opt.h, [195](#)
- API_MSG_DEBUG
 - opt.h, [195](#)
- ARP_QUEUEING
 - opt.h, [195](#)
- ARP_TABLE_SIZE
 - opt.h, [195](#)
- AUTHDEBUG
 - pppdebug.h, [307](#)
- AUTOIP_DEBUG
 - opt.h, [195](#)
- accept_local
 - ipcp_options, [23](#)
- accept_remote
 - ipcp_options, [23](#)
- ackci
 - fsm_callbacks, [15](#)
- addci
 - fsm_callbacks, [15](#)
- addr
 - ip_addr, [18](#)
 - netbuf, [29](#)
- arch.h
 - BIG_ENDIAN, [155](#)
 - LITTLE_ENDIAN, [155](#)
 - LWIP_UNUSED_ARG, [155](#)
 - PACK_STRUCT_BEGIN, [155](#)
 - PACK_STRUCT_END, [156](#)
 - PACK_STRUCT_FIELD, [156](#)
 - SZT_F, [156](#)
 - X8_F, [156](#)
- arg
 - sys_timeo, [34](#)
- asyncmap
 - lcp_options, [25](#)
- auth.h
 - auth_check_options, [284](#)
 - auth_ip_addr, [284](#)
 - auth_peer_fail, [284](#)
 - auth_peer_success, [284](#)
 - auth_reset, [284](#)
 - auth_withpeer_fail, [284](#)
 - auth_withpeer_success, [284](#)
 - bad_ip_adrs, [284](#)
 - check_passwd, [284](#)
 - get_secret, [284](#)
 - link_down, [284](#)
 - link_established, [284](#)
 - link_required, [284](#)
 - link_terminated, [284](#)
 - np_down, [284](#)
 - np_finished, [284](#)
 - np_up, [284](#)
- auth_check_options
 - auth.h, [284](#)
- auth_ip_addr
 - auth.h, [284](#)
- auth_peer_fail
 - auth.h, [284](#)
- auth_peer_success
 - auth.h, [284](#)
- auth_reset
 - auth.h, [284](#)
- auth_withpeer_fail
 - auth.h, [284](#)
- auth_withpeer_success
 - auth.h, [284](#)
- avChurnRand
 - randm.h, [310](#)
- avGenRand
 - randm.h, [310](#)
- avRandom
 - randm.h, [310](#)
- avRandomInit
 - randm.h, [310](#)
- avRandomize
 - randm.h, [310](#)
- BIG_ENDIAN
 - arch.h, [155](#)
- bad_ip_adrs
 - auth.h, [284](#)
- buf
 - MD5_CTX, [27](#)
- CBCP_OPT
 - lcp.h, [298](#)
- CHAP_CHALLENGE
 - chap.h, [286](#)
- CHAP_DIGEST_MD5
 - chap.h, [286](#)
- CHAP_FAILURE
 - chap.h, [286](#)
- CHAP_HEADERLEN
 - chap.h, [286](#)
- CHAP_MICROSOFT
 - chap.h, [286](#)
- CHAP_RESPONSE

- chap.h, [286](#)
- CHAP_SUCCESS
 - chap.h, [286](#)
- CHAPCS_CLOSED
 - chap.h, [287](#)
- CHAPCS_INITIAL
 - chap.h, [287](#)
- CHAPCS_LISTEN
 - chap.h, [287](#)
- CHAPCS_OPEN
 - chap.h, [287](#)
- CHAPCS_PENDING
 - chap.h, [287](#)
- CHAPCS_RESPONSE
 - chap.h, [287](#)
- CHAPDEBUG
 - pppdebug.h, [307](#)
- CHAPSS_BADAUTH
 - chap.h, [287](#)
- CHAPSS_CLOSED
 - chap.h, [287](#)
- CHAPSS_INITIAL
 - chap.h, [287](#)
- CHAPSS_INITIAL_CHAL
 - chap.h, [287](#)
- CHAPSS_OPEN
 - chap.h, [287](#)
- CHAPSS_PENDING
 - chap.h, [287](#)
- CHAPSS_RECHALLENGE
 - chap.h, [288](#)
- CHECKSUM_CHECK_IP
 - opt.h, [195](#)
- CHECKSUM_CHECK_TCP
 - opt.h, [195](#)
- CHECKSUM_CHECK_UDP
 - opt.h, [195](#)
- CHECKSUM_GEN_ICMP
 - opt.h, [195](#)
- CHECKSUM_GEN_IP
 - opt.h, [196](#)
- CHECKSUM_GEN_IP_INLINE
 - ip.c, [63](#)
- CHECKSUM_GEN_TCP
 - opt.h, [196](#)
- CHECKSUM_GEN_UDP
 - opt.h, [196](#)
- CI_ACCOMPRESSION
 - lcp.h, [298](#)
- CI_ADDR
 - ipcp.h, [295](#)
- CI_ADDRS
 - ipcp.h, [295](#)
- CI_ASYNCMAP
 - lcp.h, [298](#)
- CI_AUTHTYPE
 - lcp.h, [298](#)
- CI_CALLBACK
 - lcp.h, [299](#)
- CI_COMPRESSTYPE
 - ipcp.h, [295](#)
- CI_EPDISC
 - lcp.h, [299](#)
- CI_MAGICNUMBER
 - lcp.h, [299](#)
- CI_MRRU
 - lcp.h, [299](#)
- CI_MRU
 - lcp.h, [299](#)
- CI_MS_DNS1
 - ipcp.h, [295](#)
- CI_MS_DNS2
 - ipcp.h, [295](#)
- CI_MS_WINS1
 - ipcp.h, [295](#)
- CI_MS_WINS2
 - ipcp.h, [295](#)
- CI_PCOMPRESSION
 - lcp.h, [299](#)
- CI_QUALITY
 - lcp.h, [299](#)
- CI_SSNHF
 - lcp.h, [299](#)
- CODEREJ
 - fsm.h, [291](#)
- CONFACK
 - fsm.h, [291](#)
- CONFNAK
 - fsm.h, [291](#)
- CONFREJ
 - fsm.h, [292](#)
- CONFREQ
 - fsm.h, [292](#)
- callbacks
 - fsm, [13](#)
- cb
 - tcpip_msg, [35](#)
- cflag
 - ipcp_options, [23](#)
- chal_id
 - chap_state, [9](#)
- chal_interval
 - chap_state, [9](#)
- chal_len
 - chap_state, [10](#)
- chal_name
 - chap_state, [10](#)
- chal_transmits
 - chap_state, [10](#)
- chal_type
 - chap_state, [10](#)
- challenge
 - chap_state, [10](#)
- chap
 - chap.h, [288](#)
- chap.h

- CHAP_CHALLENGE, 286
- CHAP_DIGEST_MD5, 286
- CHAP_FAILURE, 286
- CHAP_HEADERLEN, 286
- CHAP_MICROSOFT, 286
- CHAP_RESPONSE, 286
- CHAP_SUCCESS, 286
- CHAPCS_CLOSED, 287
- CHAPCS_INITIAL, 287
- CHAPCS_LISTEN, 287
- CHAPCS_OPEN, 287
- CHAPCS_PENDING, 287
- CHAPCS_RESPONSE, 287
- CHAPSS_BADAUTH, 287
- CHAPSS_CLOSED, 287
- CHAPSS_INITIAL, 287
- CHAPSS_INITIAL_CHAL, 287
- CHAPSS_OPEN, 287
- CHAPSS_PENDING, 287
- CHAPSS_RECHALLENGE, 288
- chap, 288
- chap_protent, 288
- chap_state, 288
- ChapAuthPeer, 288
- ChapAuthWithPeer, 288
- MAX_CHALLENGE_LENGTH, 288
- MAX_RESPONSE_LENGTH, 288
- MD5_SIGNATURE_SIZE, 288
- MIN_CHALLENGE_LENGTH, 288
- MS_CHAP_RESPONSE_LEN, 288
- chap_mdtype
 - lcp_options, 25
- chap_protent
 - chap.h, 288
- chap_state, 9
 - chal_id, 9
 - chal_interval, 9
 - chal_len, 10
 - chal_name, 10
 - chal_transmits, 10
 - chal_type, 10
 - challenge, 10
 - chap.h, 288
 - clientstate, 10
 - id, 10
 - max_transmits, 10
 - resp_id, 10
 - resp_length, 10
 - resp_name, 10
 - resp_transmits, 10
 - resp_type, 11
 - response, 11
 - serverstate, 11
 - timeouttime, 11
 - unit, 11
- ChapAuthPeer
 - chap.h, 288
- ChapAuthWithPeer
 - chap.h, 288
- chap.h, 288
- ChapMS
 - chpms.h, 290
- check_passwd
 - auth.h, 284
- chpms.c
 - USE_CRYPT, 289
- chpms.h
 - ChapMS, 290
 - MAX_NT_PASSWORD, 290
- cilen
 - fsm_callbacks, 15
- clientstate
 - chap_state, 10
- compressSlot
 - vjcompress, 37
- cs_filler
 - cstate, 12
- cs_hdr
 - vj.h, 312
- cs_hlen
 - cstate, 12
- cs_id
 - cstate, 12
- cs_ip
 - vj.h, 312
- cs_next
 - cstate, 12
- cstate, 11
 - cs_filler, 12
 - cs_hlen, 12
 - cs_id, 12
 - cs_next, 12
 - csu_hdr, 12
 - csu_ip, 12
 - vjcs_u, 12
- csu_hdr
 - cstate, 12
- csu_ip
 - cstate, 12
- ctx
 - tcpip_msg, 35
- current_header
 - ip.c, 65
 - ipv4/lwip/ip.h, 134
- current_iphdr_dest
 - ip.c, 65
 - ipv4/lwip/ip.h, 134
- current_iphdr_src
 - ip.c, 65
 - ipv4/lwip/ip.h, 134
- current_netif
 - ip.c, 65
 - ipv4/lwip/ip.h, 134
- DEFAULT_ACCEPTMBOX_SIZE
 - opt.h, 196
- DEFAULT_RAW_RECVMBOX_SIZE
 - opt.h, 196

- DEFAULT_TCP_RECVMBOX_SIZE
 - opt.h, [196](#)
- DEFAULT_THREAD_NAME
 - opt.h, [196](#)
- DEFAULT_THREAD_PRIO
 - opt.h, [196](#)
- DEFAULT_THREAD_STACKSIZE
 - opt.h, [196](#)
- DEFAULT_UDP_RECVMBOX_SIZE
 - opt.h, [197](#)
- DEFLOOPBACKFAIL
 - lcp.h, [299](#)
- DHCP_DEBUG
 - opt.h, [197](#)
- DHCP_DOES_ARP_CHECK
 - opt.h, [197](#)
- DISCREQ
 - lcp.h, [299](#)
- DNS_DEBUG
 - opt.h, [197](#)
- DNS_DOES_NAME_CHECK
 - opt.h, [197](#)
- DNS_LOCAL_HOSTLIST
 - opt.h, [197](#)
- DNS_LOCAL_HOSTLIST_IS_DYNAMIC
 - opt.h, [197](#)
- DNS_MAX_NAME_LENGTH
 - opt.h, [197](#)
- DNS_MAX_SERVERS
 - opt.h, [198](#)
- DNS_MSG_SIZE
 - opt.h, [198](#)
- DNS_TABLE_SIZE
 - opt.h, [198](#)
- debug.h
 - LWIP_ASSERT, [157](#)
 - LWIP_DBG_FRESH, [157](#)
 - LWIP_DBG_HALT, [157](#)
 - LWIP_DBG_LEVEL_ALL, [157](#)
 - LWIP_DBG_LEVEL_OFF, [157](#)
 - LWIP_DBG_LEVEL_SERIOUS, [158](#)
 - LWIP_DBG_LEVEL_SEVERE, [158](#)
 - LWIP_DBG_LEVEL_WARNING, [158](#)
 - LWIP_DBG_MASK_LEVEL, [158](#)
 - LWIP_DBG_OFF, [158](#)
 - LWIP_DBG_ON, [158](#)
 - LWIP_DBG_STATE, [158](#)
 - LWIP_DBG_TRACE, [158](#)
 - LWIP_DEBUGF, [158](#)
 - LWIP_ERROR, [158](#)
- def.c
 - lwip_htonl, [51](#)
 - lwip_htons, [52](#)
 - lwip_ntohl, [52](#)
 - lwip_ntohs, [52](#)
- def.h
 - htonl, [160](#)
 - htons, [160](#)
 - LWIP_MAKE_U16, [160](#)
 - LWIP_MAX, [160](#)
 - LWIP_MIN, [160](#)
 - LWIP_PLATFORM_BYTESWAP, [161](#)
 - lwip_htonl, [160](#)
 - lwip_htons, [160](#)
 - lwip_ntohl, [160](#)
 - lwip_ntohs, [160](#)
 - NULL, [161](#)
 - ntohl, [161](#)
 - ntohs, [161](#)
 - PP_HTONL, [161](#)
 - PP_HTONS, [161](#)
 - PP_NTOHL, [161](#)
 - PP_NTOHS, [161](#)
- default_route
 - ipcp_options, [23](#)
- dest
 - ip_hdr, [21](#)
- digest
 - MD5_CTX, [27](#)
- dnsaddr
 - ipcp_options, [23](#)
- down
 - fsm_callbacks, [15](#)
- ECHOREP
 - lcp.h, [299](#)
- ECHOREQ
 - lcp.h, [299](#)
- ENABLE_LOOPBACK
 - netif.h, [182](#)
- ERR_ABRT
 - err.h, [165](#)
- ERR_ARG
 - err.h, [165](#)
- ERR_BUF
 - err.h, [165](#)
- ERR_CLSD
 - err.h, [165](#)
- ERR_CONN
 - err.h, [165](#)
- ERR_IF
 - err.h, [165](#)
- ERR_INPROGRESS
 - err.h, [165](#)
- ERR_IS_FATAL
 - err.h, [165](#)
- ERR_ISCONN
 - err.h, [165](#)
- ERR_MEM
 - err.h, [165](#)
- ERR_OK
 - err.h, [165](#)
- ERR_RST
 - err.h, [166](#)
- ERR_RTE
 - err.h, [166](#)
- ERR_TIMEOUT

- err.h, 166
- ERR_USE
 - err.h, 166
- ERR_VAL
 - err.h, 166
- ERR_WOULDBLOCK
 - err.h, 166
- ETH_PAD_SIZE
 - opt.h, 198
- ETHARP_DEBUG
 - opt.h, 198
- ETHARP_STATS
 - opt.h, 198
- ETHARP_STATS_DISPLAY
 - stats.h, 254
- ETHARP_STATS_INC
 - stats.h, 254
- ETHARP_SUPPORT_STATIC_ENTRIES
 - opt.h, 198
- ETHARP_SUPPORT_VLAN
 - opt.h, 198
- ETHARP_TRUST_IP_MAC
 - opt.h, 199
- err.h
 - ERR_ABRT, 165
 - ERR_ARG, 165
 - ERR_BUF, 165
 - ERR_CLSD, 165
 - ERR_CONN, 165
 - ERR_IF, 165
 - ERR_INPROGRESS, 165
 - ERR_IS_FATAL, 165
 - ERR_ISCONN, 165
 - ERR_MEM, 165
 - ERR_OK, 165
 - ERR_RST, 166
 - ERR_RTE, 166
 - ERR_TIMEOUT, 166
 - ERR_USE, 166
 - ERR_VAL, 166
 - ERR_WOULDBLOCK, 166
- err_t
 - lwip_strerror, 166
- err_t
 - err.h, 166
- extcode
 - fsm_callbacks, 16
- FOLD_U32T
 - inet_chksum.h, 124
- FSMDEBUG
 - pppdebug.h, 308
- finished
 - fsm_callbacks, 16
- flags
 - fsm, 13
 - netif, 30
 - pbuf, 33
 - vjcompress, 37
- flow1
 - ip_hdr, 21
- flow2
 - ip_hdr, 21
- fsm, 13
 - callbacks, 13
 - flags, 13
 - fsm.h, 293
 - id, 13
 - maxconfreqtransmits, 14
 - maxnakloops, 14
 - maxtermtransmits, 14
 - nakloops, 14
 - protocol, 14
 - reqid, 14
 - retransmits, 14
 - seen_ack, 14
 - state, 14
 - term_reason, 14
 - term_reason_len, 14
 - timeouttime, 14
 - unit, 15
- fsm.h
 - CODEREJ, 291
 - CONFACK, 291
 - CONFNAK, 291
 - CONFREJ, 292
 - CONFREQ, 292
 - fsm, 293
 - fsm_callbacks, 293
 - fsm_close, 293
 - fsm_init, 293
 - fsm_input, 293
 - fsm_lowerdown, 293
 - fsm_lowerup, 293
 - fsm_open, 293
 - fsm_protreject, 293
 - fsm_sdata, 293
 - HEADERLEN, 292
 - LS_ACKRCVD, 292
 - LS_ACKSENT, 292
 - LS_CLOSED, 292
 - LS_CLOSING, 292
 - LS_INITIAL, 292
 - LS_OPENED, 292
 - LS_REQSENT, 292
 - LS_STARTING, 292
 - LS_STOPPED, 292
 - LS_STOPPING, 293
 - OPT_PASSIVE, 293
 - OPT_RESTART, 293
 - OPT_SILENT, 293
 - peer_mru, 294
 - TERMACK, 293
 - TERMREQ, 293
- fsm_callbacks, 15
 - ackci, 15
 - addci, 15

- cilen, [15](#)
- down, [15](#)
- extcode, [16](#)
- finished, [16](#)
- fsm.h, [293](#)
- nakci, [16](#)
- proto_name, [16](#)
- protreject, [16](#)
- rejci, [16](#)
- reqci, [16](#)
- resetci, [16](#)
- retransmit, [16](#)
- starting, [16](#)
- up, [16](#)
- fsm_close
 - fsm.h, [293](#)
- fsm_init
 - fsm.h, [293](#)
- fsm_input
 - fsm.h, [293](#)
- fsm_lowerdown
 - fsm.h, [293](#)
- fsm_lowerup
 - fsm.h, [293](#)
- fsm_open
 - fsm.h, [293](#)
- fsm_protreject
 - fsm.h, [293](#)
- fsm_sdata
 - fsm.h, [293](#)
- function
 - tcpip_msg, [35](#)
- get_secret
 - auth.h, [284](#)
- gw
 - netif, [30](#)
- h
 - sys_timeo, [34](#)
- HEADERLEN
 - fsm.h, [292](#)
- hisaddr
 - ipcp_options, [23](#)
- hoplim
 - ip_hdr, [21](#)
- htonl
 - def.h, [160](#)
 - ipv6/lwip/inet.h, [122](#)
- htons
 - def.h, [160](#)
 - ipv6/lwip/inet.h, [122](#)
- hwaddr
 - netif, [31](#)
- hwaddr_len
 - netif, [31](#)
- i
 - MD5_CTX, [27](#)
- ICMP_DEBUG
 - opt.h, [199](#)
- ICMP_DUR
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_DUR_FRAG
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_DUR_HOST
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_DUR_NET
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_DUR_PORT
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_DUR_PROTO
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_DUR_SR
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_ECHO
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_ER
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_IR
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_IRQ
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_PP
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_RD
 - ipv4/lwip/icmp.h, [111](#)
- ICMP_SQ
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_STATS
 - opt.h, [199](#)
- ICMP_STATS_DISPLAY
 - stats.h, [254](#)
- ICMP_STATS_INC
 - stats.h, [254](#)
- ICMP_TE
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_TE_FRAG
 - ipv4/lwip/icmp.h, [113](#)
- ICMP_TE_TTL
 - ipv4/lwip/icmp.h, [113](#)
- ICMP_TS
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_TSR
 - ipv4/lwip/icmp.h, [112](#)
- ICMP_TTL
 - opt.h, [199](#)
- ICMPH_CODE
 - ipv4/lwip/icmp.h, [112](#)
- ICMPH_CODE_SET
 - ipv4/lwip/icmp.h, [112](#)
- ICMPH_TYPE
 - ipv4/lwip/icmp.h, [112](#)
- ICMPH_TYPE_SET
 - ipv4/lwip/icmp.h, [112](#)
- IGMP_DEBUG
 - opt.h, [199](#)

- IGMP_STATS
 - opt.h, [199](#)
- IGMP_STATS_DISPLAY
 - stats.h, [254](#)
- IGMP_STATS_INC
 - stats.h, [255](#)
- IN_BADCLASS
 - ipv4/lwip/inet.h, [117](#)
- IN_CLASSA
 - ipv4/lwip/inet.h, [117](#)
- IN_CLASSA_HOST
 - ipv4/lwip/inet.h, [117](#)
- IN_CLASSA_MAX
 - ipv4/lwip/inet.h, [117](#)
- IN_CLASSA_NET
 - ipv4/lwip/inet.h, [117](#)
- IN_CLASSA_NS_SHIFT
 - ipv4/lwip/inet.h, [117](#)
- IN_CLASSB
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSB_HOST
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSB_MAX
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSB_NET
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSB_NS_SHIFT
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSC
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSC_HOST
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSC_MAX
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSC_NET
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSC_NS_SHIFT
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSD
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSD_HOST
 - ipv4/lwip/inet.h, [118](#)
- IN_CLASSD_MAX
 - ipv4/lwip/inet.h, [119](#)
- IN_CLASSD_NET
 - ipv4/lwip/inet.h, [119](#)
- IN_CLASSD_NS_SHIFT
 - ipv4/lwip/inet.h, [119](#)
- IN_EXPERIMENTAL
 - ipv4/lwip/inet.h, [119](#)
- IN_LOOPBACKNET
 - ipv4/lwip/inet.h, [119](#)
- IN_MULTICAST
 - ipv4/lwip/inet.h, [119](#)
- INADDR_ANY
 - ipv4/lwip/inet.h, [119](#)
- INADDR_BROADCAST
 - ipv4/lwip/inet.h, [119](#)
- INADDR_LOOPBACK
 - ipv4/lwip/inet.h, [119](#)
- INADDR_NONE
 - ipv4/lwip/inet.h, [119](#)
- INET_DEBUG
 - opt.h, [199](#)
- IP4_ADDR
 - ipv4/lwip/ip_addr.h, [140](#)
- IP6_ADDR
 - ipv6/lwip/ip_addr.h, [150](#)
- IP_ACCEPT_LINK_LAYER_ADDRESSING
 - ip.c, [63](#)
- IP_ADDR_ANY
 - ipv4/lwip/ip_addr.h, [141](#)
 - ipv6/lwip/ip_addr.h, [150](#)
- IP_ADDR_BROADCAST
 - ipv4/lwip/ip_addr.h, [141](#)
- IP_BADCLASS
 - ipv4/lwip/ip_addr.h, [143](#)
- IP_CLASSA
 - ipv4/lwip/ip_addr.h, [143](#)
- IP_CLASSA_HOST
 - ipv4/lwip/ip_addr.h, [143](#)
- IP_CLASSA_MAX
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSA_NET
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSA_NS_SHIFT
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSB
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSB_HOST
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSB_MAX
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSB_NET
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSB_NS_SHIFT
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSC
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSC_HOST
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSC_MAX
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSC_NET
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSC_NS_SHIFT
 - ipv4/lwip/ip_addr.h, [144](#)
- IP_CLASSD
 - ipv4/lwip/ip_addr.h, [145](#)
- IP_CLASSD_HOST
 - ipv4/lwip/ip_addr.h, [145](#)
- IP_CLASSD_MAX
 - ipv4/lwip/ip_addr.h, [145](#)
- IP_CLASSD_NET
 - ipv4/lwip/ip_addr.h, [145](#)
- IP_CLASSD_NS_SHIFT
 - ipv4/lwip/ip_addr.h, [145](#)
- IP_DEBUG
 - opt.h, [199](#)
- IP_DEFAULT_TTL

- opt.h, 199
- IP_DF
 - ipv4/lwip/ip.h, 127
- IP_EXPERIMENTAL
 - ipv4/lwip/ip_addr.h, 145
- IP_FORWARD
 - opt.h, 200
- IP_FORWARD_ALLOW_TX_ON_RX_NETIF
 - opt.h, 200
- IP_FRAG
 - opt.h, 200
- IP_FRAG_USES_STATIC_BUF
 - opt.h, 200
- IP_HDRINCL
 - ipv4/lwip/ip.h, 128
 - ipv6/lwip/ip.h, 136
- IP_HLEN
 - ipv4/lwip/ip.h, 128
 - ipv6/lwip/ip.h, 136
- IP_LOOPBACKNET
 - ipv4/lwip/ip_addr.h, 145
- IP_MF
 - ipv4/lwip/ip.h, 128
- IP_MULTICAST
 - ipv4/lwip/ip_addr.h, 145
- IP_OFFMASK
 - ipv4/lwip/ip.h, 128
- IP_OPTIONS_ALLOWED
 - opt.h, 200
- IP_OPTIONS_SEND
 - ipv4/lwip/ip.h, 128
- IP_PCB
 - ip_pcb, 22
 - ipv4/lwip/ip.h, 128
 - ipv6/lwip/ip.h, 136
- IP_PCB_ADDRHINT
 - ipv4/lwip/ip.h, 128
 - ipv6/lwip/ip.h, 136
- IP_PROTO_ICMP
 - ipv4/lwip/ip.h, 128
 - ipv6/lwip/ip.h, 137
- IP_PROTO_IGMP
 - ipv4/lwip/ip.h, 129
- IP_PROTO_TCP
 - ipv4/lwip/ip.h, 129
 - ipv6/lwip/ip.h, 137
- IP_PROTO_UDP
 - ipv4/lwip/ip.h, 129
 - ipv6/lwip/ip.h, 137
- IP_PROTO_UDPLITE
 - ipv4/lwip/ip.h, 129
 - ipv6/lwip/ip.h, 137
- IP_REASS_DEBUG
 - opt.h, 200
- IP_REASS_MAX_PBUFS
 - opt.h, 200
- IP_REASS_MAXAGE
 - opt.h, 201
- IP_REASSEMBLY
 - opt.h, 201
- IP_RF
 - ipv4/lwip/ip.h, 129
- IP_SOF_BROADCAST
 - opt.h, 201
- IP_SOF_BROADCAST_RECV
 - opt.h, 201
- IP_STATS
 - opt.h, 201
- IP_STATS_DISPLAY
 - stats.h, 255
- IP_STATS_INC
 - stats.h, 255
- IPADDR2_COPY
 - ipv4/lwip/ip_addr.h, 145
- IPADDR_ANY
 - ipv4/lwip/ip_addr.h, 145
- IPADDR_BROADCAST
 - ipv4/lwip/ip_addr.h, 145
- IPADDR_LOOPBACK
 - ipv4/lwip/ip_addr.h, 146
- IPADDR_NONE
 - ipv4/lwip/ip_addr.h, 146
- IPCP_VJ_COMP
 - ipcp.h, 295
- IPCP_VJ_COMP_OLD
 - ipcp.h, 296
- IPCP_VJMODE_OLD
 - ipcp.h, 296
- IPCP_VJMODE_RFC1172
 - ipcp.h, 296
- IPCP_VJMODE_RFC1332
 - ipcp.h, 296
- IPCPDEBUG
 - pppdebug.h, 308
- IPFRAG_STATS
 - opt.h, 201
- IPFRAG_STATS_DISPLAY
 - stats.h, 255
- IPFRAG_STATS_INC
 - stats.h, 255
- IPH_CHKSUM
 - ipv4/lwip/ip.h, 129
- IPH_CHKSUM_SET
 - ipv4/lwip/ip.h, 129
- IPH_HL
 - ipv4/lwip/ip.h, 129
- IPH_ID
 - ipv4/lwip/ip.h, 129
- IPH_ID_SET
 - ipv4/lwip/ip.h, 129
- IPH_LEN
 - ipv4/lwip/ip.h, 130
- IPH_LEN_SET
 - ipv4/lwip/ip.h, 130
- IPH_OFFSET
 - ipv4/lwip/ip.h, 130

- IPH_OFFSET_SET
 - ipv4/lwip/ip.h, [130](#)
- IPH_PROTO
 - ipv4/lwip/ip.h, [130](#)
 - ipv6/lwip/ip.h, [137](#)
- IPH_PROTO_SET
 - ipv4/lwip/ip.h, [130](#)
- IPH_TOS
 - ipv4/lwip/ip.h, [130](#)
- IPH_TOS_SET
 - ipv4/lwip/ip.h, [130](#)
- IPH_TTL
 - ipv4/lwip/ip.h, [130](#)
- IPH_TTL_SET
 - ipv4/lwip/ip.h, [130](#)
- IPH_V
 - ipv4/lwip/ip.h, [130](#)
- IPH_VHL_SET
 - ipv4/lwip/ip.h, [130](#)
- icmp_dur_type
 - ipv4/lwip/icmp.h, [112](#)
- icmp_echo_hdr, [17](#)
 - PACK_STRUCT_FIELD, [17](#)
- icmp_te_type
 - ipv4/lwip/icmp.h, [112](#)
- id
 - chap_state, [10](#)
 - fsm, [13](#)
- in
 - MD5_CTX, [27](#)
- in_addr, [17](#)
 - s_addr, [18](#)
- in_range
 - ip_addr.c, [67](#)
- inet6.c
 - inet_chksum, [72](#)
 - inet_chksum_pbuf, [72](#)
 - inet_chksum_pseudo, [73](#)
- inet_addr
 - ipv4/lwip/inet.h, [119](#)
 - ipv6/lwip/inet.h, [122](#)
- inet_addr_from_ipaddr
 - ipv4/lwip/inet.h, [120](#)
- inet_addr_to_ipaddr
 - ipv4/lwip/inet.h, [120](#)
- inet_addr_to_ipaddr_p
 - ipv4/lwip/inet.h, [120](#)
- inet_aton
 - ipv4/lwip/inet.h, [120](#)
 - ipv6/lwip/inet.h, [122](#)
- inet_chksum
 - inet6.c, [72](#)
 - inet_chksum.c, [61](#)
 - inet_chksum.h, [124](#)
 - ipv6/lwip/inet.h, [122](#)
- inet_chksum.c
 - inet_chksum, [61](#)
 - inet_chksum_pbuf, [61](#)
 - inet_chksum_pseudo, [61](#)
 - inet_chksum_pseudo_partial, [61](#)
 - LWIP_CHKSUM, [60](#)
 - LWIP_CHKSUM_ALGORITHM, [61](#)
- inet_chksum.h
 - FOLD_U32T, [124](#)
 - inet_chksum, [124](#)
 - inet_chksum_pbuf, [124](#)
 - inet_chksum_pseudo, [124](#)
 - inet_chksum_pseudo_partial, [124](#)
 - LWIP_CHKSUM_COPY_ALGORITHM, [124](#)
 - SWAP_BYTES_IN_WORD, [124](#)
- inet_chksum_pbuf
 - inet6.c, [72](#)
 - inet_chksum.c, [61](#)
 - inet_chksum.h, [124](#)
 - ipv6/lwip/inet.h, [122](#)
- inet_chksum_pseudo
 - inet6.c, [73](#)
 - inet_chksum.c, [61](#)
 - inet_chksum.h, [124](#)
 - ipv6/lwip/inet.h, [122](#)
- inet_chksum_pseudo_partial
 - inet_chksum.c, [61](#)
 - inet_chksum.h, [124](#)
- inet_ntoa
 - ipv4/lwip/inet.h, [120](#)
- inet_ntoa_r
 - ipv4/lwip/inet.h, [120](#)
- init.c
 - LWIP_DISABLE_MEMP_SANITY_CHECKS, [55](#)
 - LWIP_DISABLE_TCP_SANITY_CHECKS, [55](#)
 - lwip_init, [56](#)
- init.h
 - LWIP_RC_DEVELOPMENT, [168](#)
 - LWIP_RC_RELEASE, [168](#)
 - LWIP_VERSION, [168](#)
 - LWIP_VERSION_IS_DEVELOPMENT, [168](#)
 - LWIP_VERSION_IS_RC, [168](#)
 - LWIP_VERSION_IS_RELEASE, [168](#)
 - LWIP_VERSION_MAJOR, [168](#)
 - LWIP_VERSION_MINOR, [168](#)
 - LWIP_VERSION_RC, [169](#)
 - LWIP_VERSION_REVISION, [169](#)
 - lwip_init, [169](#)
- inp
 - tcpip_msg, [36](#)
- input
 - netif, [31](#)
- ip.c
 - CHECKSUM_GEN_IP_INLINE, [63](#)
 - current_header, [65](#)
 - current_iphdr_dest, [65](#)
 - current_iphdr_src, [65](#)
 - current_netif, [65](#)
 - IP_ACCEPT_LINK_LAYER_ADDRESSING, [63](#)
 - ip_input, [63](#)
 - ip_output, [63](#)

- ip_output_if, 64
 - ip_route, 64
 - LWIP_INLINE_IP_CHKSUM, 63
- ip4_addr1
 - ipv4/lwip/ip_addr.h, 140
- ip4_addr1_16
 - ipv4/lwip/ip_addr.h, 140
- ip4_addr2
 - ipv4/lwip/ip_addr.h, 140
- ip4_addr2_16
 - ipv4/lwip/ip_addr.h, 140
- ip4_addr3
 - ipv4/lwip/ip_addr.h, 141
- ip4_addr3_16
 - ipv4/lwip/ip_addr.h, 141
- ip4_addr4
 - ipv4/lwip/ip_addr.h, 141
- ip4_addr4_16
 - ipv4/lwip/ip_addr.h, 141
- ip4_addr_get_u32
 - ipv4/lwip/ip_addr.h, 141
- ip4_addr_isbroadcast
 - ip_addr.c, 67
 - ipv4/lwip/ip_addr.h, 146
- ip4_addr_netmask_valid
 - ip_addr.c, 67
 - ipv4/lwip/ip_addr.h, 146
- ip4_addr_set_u32
 - ipv4/lwip/ip_addr.h, 141
- ip6.c
 - ip_init, 74
 - ip_input, 74
 - ip_output, 74
 - ip_output_if, 74
 - ip_route, 74
- ip6_addr.c
 - ip_addr_cmp, 75
 - ip_addr_isany, 75
 - ip_addr_netcmp, 75
 - ip_addr_set, 75
- ip_addr, 18
 - addr, 18
 - netif, 31
 - PACK_STRUCT_FIELD, 18
- ip_addr.c
 - in_range, 67
 - ip4_addr_isbroadcast, 67
 - ip4_addr_netmask_valid, 67
 - ip_addr_any, 70
 - ip_addr_broadcast, 70
 - ipaddr_addr, 69
 - ipaddr_aton, 69
 - ipaddr_ntoa, 69
 - ipaddr_ntoa_r, 69
 - isdigit, 67
 - islower, 67
 - isprint, 67
 - isspace, 67
 - isxdigit, 67
- ip_addr2, 18
 - PACK_STRUCT_FIELD, 19
- ip_addr_any
 - ip_addr.c, 70
 - ipv4/lwip/ip_addr.h, 148
- ip_addr_broadcast
 - ip_addr.c, 70
 - ipv4/lwip/ip_addr.h, 148
- ip_addr_cmp
 - ip6_addr.c, 75
 - ipv4/lwip/ip_addr.h, 141
 - ipv6/lwip/ip_addr.h, 150
- ip_addr_copy
 - ipv4/lwip/ip_addr.h, 141
- ip_addr_debug_print
 - ipv4/lwip/ip_addr.h, 141
 - ipv6/lwip/ip_addr.h, 150
- ip_addr_get_network
 - ipv4/lwip/ip_addr.h, 142
- ip_addr_isany
 - ip6_addr.c, 75
 - ipv4/lwip/ip_addr.h, 142
 - ipv6/lwip/ip_addr.h, 150
- ip_addr_isbroadcast
 - ipv4/lwip/ip_addr.h, 142
- ip_addr_islinklocal
 - ipv4/lwip/ip_addr.h, 142
- ip_addr_ismulticast
 - ipv4/lwip/ip_addr.h, 142
- ip_addr_netcmp
 - ip6_addr.c, 75
 - ipv4/lwip/ip_addr.h, 142
 - ipv6/lwip/ip_addr.h, 150
- ip_addr_netmask_valid
 - ipv4/lwip/ip_addr.h, 142
- ip_addr_p_t
 - ipv4/lwip/ip_addr.h, 146
- ip_addr_packed, 19
 - PACK_STRUCT_FIELD, 19
- ip_addr_set
 - ip6_addr.c, 75
 - ipv4/lwip/ip_addr.h, 143
 - ipv6/lwip/ip_addr.h, 151
- ip_addr_set_any
 - ipv4/lwip/ip_addr.h, 143
- ip_addr_set_hton
 - ipv4/lwip/ip_addr.h, 143
- ip_addr_set_loopback
 - ipv4/lwip/ip_addr.h, 143
- ip_addr_set_zero
 - ipv4/lwip/ip_addr.h, 143
- ip_addr_t
 - ipv4/lwip/ip_addr.h, 146
- ip_current_dest_addr
 - ipv4/lwip/ip.h, 127
- ip_current_header
 - ipv4/lwip/ip.h, 127

- ipv6/lwip/ip.h, 136
- ip_current_netif
 - ipv4/lwip/ip.h, 127
 - ipv6/lwip/ip.h, 136
- ip_current_src_addr
 - ipv4/lwip/ip.h, 127
- ip_debug_print
 - ipv4/lwip/ip.h, 127
- ip_get_option
 - ipv4/lwip/ip.h, 127
- ip_hdr, 19
 - dest, 21
 - flow1, 21
 - flow2, 21
 - hoplim, 21
 - len, 21
 - nexthdr, 21
 - PACK_STRUCT_FIELD, 20, 21
 - tclass1, 21
 - tclass2, 21
 - v, 21
- ip_init
 - ip6.c, 74
 - ipv4/lwip/ip.h, 128
 - ipv6/lwip/ip.h, 137
- ip_input
 - ip.c, 63
 - ip6.c, 74
 - ipv4/lwip/ip.h, 131
 - ipv6/lwip/ip.h, 137
- ip_ntoa
 - ipv4/lwip/ip_addr.h, 145
- ip_output
 - ip.c, 63
 - ip6.c, 74
 - ipv4/lwip/ip.h, 131
 - ipv6/lwip/ip.h, 137
- ip_output_if
 - ip.c, 64
 - ip6.c, 74
 - ipv4/lwip/ip.h, 133
 - ipv6/lwip/ip.h, 137
- ip_pcb, 22
 - IP_PCB, 22
- ip_reset_option
 - ipv4/lwip/ip.h, 129
- ip_route
 - ip.c, 64
 - ip6.c, 74
 - ipv4/lwip/ip.h, 133
 - ipv6/lwip/ip.h, 138
- ip_set_option
 - ipv4/lwip/ip.h, 129
- ipaddr_addr
 - ip_addr.c, 69
 - ipv4/lwip/ip_addr.h, 147
- ipaddr_aton
 - ip_addr.c, 69
- ipv4/lwip/ip.h, 136
- ipaddr_ntoa
 - ip_addr.c, 69
 - ipv4/lwip/ip_addr.h, 147
- ipaddr_ntoa_r
 - ip_addr.c, 69
 - ipv4/lwip/ip_addr.h, 147
- ipcp.h
 - CI_ADDR, 295
 - CI_ADDRS, 295
 - CI_COMPRESSTYPE, 295
 - CI_MS_DNS1, 295
 - CI_MS_DNS2, 295
 - CI_MS_WINS1, 295
 - CI_MS_WINS2, 295
 - IPCP_VJ_COMP, 295
 - IPCP_VJ_COMP_OLD, 296
 - IPCP_VJMODE_OLD, 296
 - IPCP_VJMODE_RFC1172, 296
 - IPCP_VJMODE_RFC1332, 296
 - ipcp_allowoptions, 296
 - ipcp_fsm, 296
 - ipcp_gotoptions, 296
 - ipcp_hisoptions, 296
 - ipcp_options, 296
 - ipcp_protent, 296
 - ipcp_wantoptions, 296
- ipcp_allowoptions
 - ipcp.h, 296
- ipcp_fsm
 - ipcp.h, 296
- ipcp_gotoptions
 - ipcp.h, 296
- ipcp_hisoptions
 - ipcp.h, 296
- ipcp_options, 22
 - accept_local, 23
 - accept_remote, 23
 - cflag, 23
 - default_route, 23
 - dnsaddr, 23
 - hisaddr, 23
 - ipcp.h, 296
 - maxslotindex, 23
 - neg_addr, 23
 - neg_vj, 23
 - old_addrs, 23
 - old_vj, 23
 - ouraddr, 24
 - proxy_arp, 24
 - req_addr, 24
 - req_dns1, 24
 - req_dns2, 24
 - vj_protocol, 24
 - winsaddr, 24
- ipcp_protent
 - ipcp.h, 296
- ipcp_wantoptions

- ipcp.h, 296
- ipv4/lwip/icmp.h
 - ICMP_DUR, 111
 - ICMP_DUR_FRAG, 112
 - ICMP_DUR_HOST, 112
 - ICMP_DUR_NET, 112
 - ICMP_DUR_PORT, 112
 - ICMP_DUR_PROTO, 112
 - ICMP_DUR_SR, 112
 - ICMP_ECHO, 111
 - ICMP_ER, 111
 - ICMP_IR, 111
 - ICMP_IRQ, 111
 - ICMP_PP, 111
 - ICMP_RD, 111
 - ICMP_SQ, 112
 - ICMP_TE, 112
 - ICMP_TE_FRAG, 113
 - ICMP_TE_TTL, 113
 - ICMP_TS, 112
 - ICMP_TSR, 112
 - ICMPH_CODE, 112
 - ICMPH_CODE_SET, 112
 - ICMPH_TYPE, 112
 - ICMPH_TYPE_SET, 112
 - icmp_dur_type, 112
 - icmp_te_type, 112
 - PACK_STRUCT_STRUCT, 113
- ipv4/lwip/inet.h
 - IN_BADCLASS, 117
 - IN_CLASSA, 117
 - IN_CLASSA_HOST, 117
 - IN_CLASSA_MAX, 117
 - IN_CLASSA_NET, 117
 - IN_CLASSA_NS_SHIFT, 117
 - IN_CLASSB, 118
 - IN_CLASSB_HOST, 118
 - IN_CLASSB_MAX, 118
 - IN_CLASSB_NET, 118
 - IN_CLASSB_NS_SHIFT, 118
 - IN_CLASSC, 118
 - IN_CLASSC_HOST, 118
 - IN_CLASSC_MAX, 118
 - IN_CLASSC_NET, 118
 - IN_CLASSC_NS_SHIFT, 118
 - IN_CLASSD, 118
 - IN_CLASSD_HOST, 118
 - IN_CLASSD_MAX, 119
 - IN_CLASSD_NET, 119
 - IN_CLASSD_NS_SHIFT, 119
 - IN_EXPERIMENTAL, 119
 - IN_LOOPBACKNET, 119
 - IN_MULTICAST, 119
 - INADDR_ANY, 119
 - INADDR_BROADCAST, 119
 - INADDR_LOOPBACK, 119
 - INADDR_NONE, 119
 - inet_addr, 119
 - inet_addr_from_ipaddr, 120
 - inet_addr_to_ipaddr, 120
 - inet_addr_to_ipaddr_p, 120
 - inet_aton, 120
 - inet_ntoa, 120
 - inet_ntoa_r, 120
- ipv4/lwip/ip.h
 - current_header, 134
 - current_iphdr_dest, 134
 - current_iphdr_src, 134
 - current_netif, 134
 - IP_DF, 127
 - IP_HDRINCL, 128
 - IP_HLEN, 128
 - IP_MF, 128
 - IP_OFFMASK, 128
 - IP_OPTIONS_SEND, 128
 - IP_PCB, 128
 - IP_PCB_ADDRHINT, 128
 - IP_PROTO_ICMP, 128
 - IP_PROTO_IGMP, 129
 - IP_PROTO_TCP, 129
 - IP_PROTO_UDP, 129
 - IP_PROTO_UDPLITE, 129
 - IP_RF, 129
 - IPH_CHKSUM, 129
 - IPH_CHKSUM_SET, 129
 - IPH_HL, 129
 - IPH_ID, 129
 - IPH_ID_SET, 129
 - IPH_LEN, 130
 - IPH_LEN_SET, 130
 - IPH_OFFSET, 130
 - IPH_OFFSET_SET, 130
 - IPH_PROTO, 130
 - IPH_PROTO_SET, 130
 - IPH_TOS, 130
 - IPH_TOS_SET, 130
 - IPH_TTL, 130
 - IPH_TTL_SET, 130
 - IPH_V, 130
 - IPH_VHL_SET, 130
 - ip_current_dest_addr, 127
 - ip_current_header, 127
 - ip_current_netif, 127
 - ip_current_src_addr, 127
 - ip_debug_print, 127
 - ip_get_option, 127
 - ip_init, 128
 - ip_input, 131
 - ip_output, 131
 - ip_output_if, 133
 - ip_reset_option, 129
 - ip_route, 133
 - ip_set_option, 129
 - PACK_STRUCT_STRUCT, 134
 - SOF_ACCEPTCONN, 131
 - SOF_BROADCAST, 131

- SOF_INHERITED, [131](#)
- SOF_KEEPALIVE, [131](#)
- SOF_LINGER, [131](#)
- SOF_REUSEADDR, [131](#)
- ipv4/lwip/ip_addr.h
 - IP4_ADDR, [140](#)
 - IP_ADDR_ANY, [141](#)
 - IP_ADDR_BROADCAST, [141](#)
 - IP_BADCLASS, [143](#)
 - IP_CLASSA, [143](#)
 - IP_CLASSA_HOST, [143](#)
 - IP_CLASSA_MAX, [144](#)
 - IP_CLASSA_NET, [144](#)
 - IP_CLASSA_NSIFT, [144](#)
 - IP_CLASSB, [144](#)
 - IP_CLASSB_HOST, [144](#)
 - IP_CLASSB_MAX, [144](#)
 - IP_CLASSB_NET, [144](#)
 - IP_CLASSB_NSIFT, [144](#)
 - IP_CLASSC, [144](#)
 - IP_CLASSC_HOST, [144](#)
 - IP_CLASSC_NET, [144](#)
 - IP_CLASSC_NSIFT, [144](#)
 - IP_CLASSD, [145](#)
 - IP_CLASSD_HOST, [145](#)
 - IP_CLASSD_NET, [145](#)
 - IP_CLASSD_NSIFT, [145](#)
 - IP_EXPERIMENTAL, [145](#)
 - IP_LOOPBACKNET, [145](#)
 - IP_MULTICAST, [145](#)
 - IPADDR2_COPY, [145](#)
 - IPADDR_ANY, [145](#)
 - IPADDR_BROADCAST, [145](#)
 - IPADDR_LOOPBACK, [146](#)
 - IPADDR_NONE, [146](#)
 - ip4_addr1, [140](#)
 - ip4_addr1_16, [140](#)
 - ip4_addr2, [140](#)
 - ip4_addr2_16, [140](#)
 - ip4_addr3, [141](#)
 - ip4_addr3_16, [141](#)
 - ip4_addr4, [141](#)
 - ip4_addr4_16, [141](#)
 - ip4_addr_get_u32, [141](#)
 - ip4_addr_isbroadcast, [146](#)
 - ip4_addr_netmask_valid, [146](#)
 - ip4_addr_set_u32, [141](#)
 - ip_addr_any, [148](#)
 - ip_addr_broadcast, [148](#)
 - ip_addr_cmp, [141](#)
 - ip_addr_copy, [141](#)
 - ip_addr_debug_print, [141](#)
 - ip_addr_get_network, [142](#)
 - ip_addr_isany, [142](#)
 - ip_addr_isbroadcast, [142](#)
 - ip_addr_islinklocal, [142](#)
 - ip_addr_ismulticast, [142](#)
 - ip_addr_netcmp, [142](#)
 - ip_addr_netmask_valid, [142](#)
 - ip_addr_p_t, [146](#)
 - ip_addr_set, [143](#)
 - ip_addr_set_any, [143](#)
 - ip_addr_set_hton, [143](#)
 - ip_addr_set_loopback, [143](#)
 - ip_addr_set_zero, [143](#)
 - ip_addr_t, [146](#)
 - ip_ntoa, [145](#)
 - ipaddr_addr, [147](#)
 - ipaddr_aton, [147](#)
 - ipaddr_ntoa, [147](#)
 - ipaddr_ntoa_r, [147](#)
 - PACK_STRUCT_STRUCT, [148](#)
- ipv6/lwip/inet.h
 - htonl, [122](#)
 - htons, [122](#)
 - inet_addr, [122](#)
 - inet_aton, [122](#)
 - inet_chksum, [122](#)
 - inet_chksum_pbuf, [122](#)
 - inet_chksum_pseudo, [122](#)
 - ntohl, [122](#)
 - ntohs, [122](#)
- ipv6/lwip/ip.h
 - IP_HDRINCL, [136](#)
 - IP_HLEN, [136](#)
 - IP_PCB, [136](#)
 - IP_PCB_ADDRHINT, [136](#)
 - IP_PROTO_ICMP, [137](#)
 - IP_PROTO_TCP, [137](#)
 - IP_PROTO_UDP, [137](#)
 - IP_PROTO_UDPLITE, [137](#)
 - IPH_PROTO, [137](#)
 - ip_current_header, [136](#)
 - ip_current_netif, [136](#)
 - ip_init, [137](#)
 - ip_input, [137](#)
 - ip_output, [137](#)
 - ip_output_if, [137](#)
 - ip_route, [138](#)
- ipv6/lwip/ip_addr.h
 - IP6_ADDR, [150](#)
 - IP_ADDR_ANY, [150](#)
 - ip_addr_cmp, [150](#)
 - ip_addr_debug_print, [150](#)
 - ip_addr_isany, [150](#)
 - ip_addr_netcmp, [150](#)
 - ip_addr_set, [151](#)
 - PACK_STRUCT_STRUCT, [151](#)
- isdigit
 - ip_addr.c, [67](#)
- islower
 - ip_addr.c, [67](#)
- isprint
 - ip_addr.c, [67](#)
- isspace
 - ip_addr.c, [67](#)

- isxdigit
 - ip_addr.c, 67
- LCPDEBUG
 - pppdebug.h, 308
- LINK_STATS
 - opt.h, 201
- LINK_STATS_DISPLAY
 - stats.h, 255
- LINK_STATS_INC
 - stats.h, 255
- LITTLE_ENDIAN
 - arch.h, 155
- LOCK_TCPIP_CORE
 - tcpip.h, 269
- LOG_CRITICAL
 - pppdebug.h, 308
- LOG_DEBUG
 - pppdebug.h, 308
- LOG_DETAIL
 - pppdebug.h, 308
- LOG_ERR
 - pppdebug.h, 308
- LOG_INFO
 - pppdebug.h, 308
- LOG_NOTICE
 - pppdebug.h, 308
- LOG_WARNING
 - pppdebug.h, 308
- LS_ACKRCVD
 - fsm.h, 292
- LS_ACKSENT
 - fsm.h, 292
- LS_CLOSED
 - fsm.h, 292
- LS_CLOSING
 - fsm.h, 292
- LS_INITIAL
 - fsm.h, 292
- LS_OPENED
 - fsm.h, 292
- LS_REQSENT
 - fsm.h, 292
- LS_STARTING
 - fsm.h, 292
- LS_STOPPED
 - fsm.h, 292
- LS_STOPPING
 - fsm.h, 293
- LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT
 - opt.h, 201
- LWIP_ARP
 - opt.h, 202
- LWIP_ASSERT
 - debug.h, 157
- LWIP_AUTOIP
 - opt.h, 202
- LWIP_BROADCAST_PING
 - opt.h, 202
- LWIP_CALLBACK_API
 - opt.h, 202
- LWIP_CHECKSUM_ON_COPY
 - opt.h, 202
- LWIP_CHKSUM
 - inet_chksum.c, 60
- LWIP_CHKSUM_ALGORITHM
 - inet_chksum.c, 61
- LWIP_CHKSUM_COPY_ALGORITHM
 - inet_chksum.h, 124
- LWIP_COMPAT_SOCKETS
 - opt.h, 202
- LWIP_DBG_FRESH
 - debug.h, 157
- LWIP_DBG_HALT
 - debug.h, 157
- LWIP_DBG_LEVEL_ALL
 - debug.h, 157
- LWIP_DBG_LEVEL_OFF
 - debug.h, 157
- LWIP_DBG_LEVEL_SERIOUS
 - debug.h, 158
- LWIP_DBG_LEVEL_SEVERE
 - debug.h, 158
- LWIP_DBG_LEVEL_WARNING
 - debug.h, 158
- LWIP_DBG_MASK_LEVEL
 - debug.h, 158
- LWIP_DBG_MIN_LEVEL
 - opt.h, 202
- LWIP_DBG_OFF
 - debug.h, 158
- LWIP_DBG_ON
 - debug.h, 158
- LWIP_DBG_STATE
 - debug.h, 158
- LWIP_DBG_TRACE
 - debug.h, 158
- LWIP_DBG_TYPES_ON
 - opt.h, 203
- LWIP_DEBUG_TIMERNAMES
 - timers.h, 274
- LWIP_DEBUGF
 - debug.h, 158
- LWIP_DHCP
 - opt.h, 203
- LWIP_DHCP_AUTOIP_COOP
 - opt.h, 203
- LWIP_DHCP_AUTOIP_COOP_TRIES
 - opt.h, 203
- LWIP_DISABLE_MEMP_SANITY_CHECKS
 - init.c, 55
- LWIP_DISABLE_TCP_SANITY_CHECKS
 - init.c, 55
- LWIP_DNS
 - opt.h, 203
- LWIP_ERROR

- debug.h, [158](#)
- LWIP_ETHERNET
 - opt.h, [203](#)
- LWIP_EVENT_API
 - opt.h, [203](#)
- LWIP_HAVE_LOOPIF
 - opt.h, [203](#)
- LWIP_HAVE_SLIPIF
 - opt.h, [204](#)
- LWIP_ICMP
 - opt.h, [204](#)
- LWIP_IGMP
 - opt.h, [204](#)
- LWIP_INLINE_IP_CHKSUM
 - ip.c, [63](#)
- LWIP_LOOPBACK_MAX_PBUFS
 - opt.h, [204](#)
- LWIP_MAKE_U16
 - def.h, [160](#)
- LWIP_MALLOC_MEMPOOL
 - memp_std.h, [175](#)
- LWIP_MALLOC_MEMPOOL_END
 - memp_std.h, [175](#)
- LWIP_MALLOC_MEMPOOL_START
 - memp_std.h, [175](#)
- LWIP_MAX
 - def.h, [160](#)
- LWIP_MEM_ALIGN
 - mem.h, [171](#)
- LWIP_MEM_ALIGN_BUFFER
 - mem.h, [171](#)
- LWIP_MEM_ALIGN_SIZE
 - mem.h, [171](#)
- LWIP_MEM_ALLOC_DECL_PROTECT
 - mem.c, [77](#)
- LWIP_MEM_ALLOC_PROTECT
 - mem.c, [77](#)
- LWIP_MEM_ALLOC_UNPROTECT
 - mem.c, [77](#)
- LWIP_MEM_FREE_DECL_PROTECT
 - mem.c, [77](#)
- LWIP_MEM_FREE_PROTECT
 - mem.c, [77](#)
- LWIP_MEM_FREE_UNPROTECT
 - mem.c, [77](#)
- LWIP_MEMPOOL
 - memp.c, [81](#)
 - memp.h, [174](#)
- LWIP_MIN
 - def.h, [160](#)
- LWIP_MULTICAST_PING
 - opt.h, [204](#)
- LWIP_NETBUF_RECVINFO
 - opt.h, [204](#)
- LWIP_NETCONN
 - opt.h, [204](#)
- LWIP_NETIF_API
 - opt.h, [204](#)
- LWIP_NETIF_HOSTNAME
 - opt.h, [204](#)
- LWIP_NETIF_HWADDRHINT
 - opt.h, [205](#)
- LWIP_NETIF_LINK_CALLBACK
 - opt.h, [205](#)
- LWIP_NETIF_LOOPBACK
 - opt.h, [205](#)
- LWIP_NETIF_LOOPBACK_MULTITHREADING
 - opt.h, [205](#)
- LWIP_NETIF_REMOVE_CALLBACK
 - opt.h, [205](#)
- LWIP_NETIF_STATUS_CALLBACK
 - opt.h, [205](#)
- LWIP_NETIF_TX_SINGLE_PBUF
 - opt.h, [205](#)
- LWIP_PBUF_MEMPOOL
 - memp_std.h, [175](#)
- LWIP_PLATFORM_BYTESWAP
 - def.h, [161](#)
- LWIP_POSIX_SOCKETS_IO_NAMES
 - opt.h, [206](#)
- LWIP_RAM_HEAP_POINTER
 - mem.c, [77](#)
- LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS
 - opt.h, [206](#)
- LWIP_RAW
 - opt.h, [206](#)
- LWIP_RC_DEVELOPMENT
 - init.h, [168](#)
- LWIP_RC_RELEASE
 - init.h, [168](#)
- LWIP_SNMP
 - opt.h, [206](#)
- LWIP_SO_RCVBUF
 - opt.h, [206](#)
- LWIP_SO_RCVTIMEO
 - opt.h, [206](#)
- LWIP_SO_SNDTIMEO
 - opt.h, [206](#)
- LWIP_SOCKET
 - opt.h, [206](#)
- LWIP_STATS
 - opt.h, [206](#)
- LWIP_STATS_DISPLAY
 - opt.h, [207](#)
- LWIP_SUPPORT_CUSTOM_PBUF
 - pbuf.h, [223](#)
- LWIP_TCP
 - opt.h, [207](#)
- LWIP_TCP_KEEPALIVE
 - opt.h, [207](#)
- LWIP_TCP_TIMESTAMPS
 - opt.h, [207](#)
- LWIP_TCPIP_CORE_LOCKING
 - opt.h, [207](#)
- LWIP_TCPIP_CORE_LOCKING_INPUT
 - opt.h, [207](#)

- LWIP_TCPIP_THREAD_ALIVE
 - tcpip.h, 269
- LWIP_TCPIP_TIMEOUT
 - opt.h, 207
- LWIP_TIMERS
 - timers.h, 274
- LWIP_UDP
 - opt.h, 207
- LWIP_UDPLITE
 - opt.h, 207
- LWIP_UNUSED_ARG
 - arch.h, 155
- LWIP_VERSION
 - init.h, 168
- LWIP_VERSION_IS_DEVELOPMENT
 - init.h, 168
- LWIP_VERSION_IS_RC
 - init.h, 168
- LWIP_VERSION_IS_RELEASE
 - init.h, 168
- LWIP_VERSION_MAJOR
 - init.h, 168
- LWIP_VERSION_MINOR
 - init.h, 168
- LWIP_VERSION_RC
 - init.h, 169
- LWIP_VERSION_REVISION
 - init.h, 169
- last_cs
 - vjcompress, 37
- last_recv
 - vjcompress, 38
- last_xmit
 - vjcompress, 38
- lcp.h
 - CBCP_OPT, 298
 - CI_ACCOMPRESSION, 298
 - CI_ASYNCMAP, 298
 - CI_AUTHTYPE, 298
 - CI_CALLBACK, 299
 - CI_EPDISC, 299
 - CI_MAGICNUMBER, 299
 - CI_MRRU, 299
 - CI_MRU, 299
 - CI_PCOMPRESSION, 299
 - CI_QUALITY, 299
 - CI_SSNHF, 299
 - DEFLOOPBACKFAIL, 299
 - DISCREQ, 299
 - ECHOREP, 299
 - ECHOREQ, 299
 - lcp_allowoptions, 300
 - lcp_close, 300
 - lcp_gotoptions, 300
 - lcp_hisoptions, 300
 - lcp_init, 300
 - lcp_lowerdown, 300
 - lcp_lowerup, 300
 - lcp_open, 300
 - lcp_options, 300
 - lcp_phase, 300
 - lcp_protent, 300
 - lcp_sprotrej, 300
 - lcp_wantoptions, 300
 - LinkPhase, 300
 - PHASE_AUTHENTICATE, 300
 - PHASE_CALLBACK, 300
 - PHASE_DEAD, 300
 - PHASE_ESTABLISH, 300
 - PHASE_INITIALIZE, 300
 - PHASE_NETWORK, 300
 - PHASE_TERMINATE, 300
 - PROTREJ, 300
 - xmit_accm, 301

- lcp_wantoptions
 - lcp.h, [300](#)
- len
 - ip_hdr, [21](#)
 - pbuf, [33](#)
- link_down
 - auth.h, [284](#)
- link_established
 - auth.h, [284](#)
- link_required
 - auth.h, [284](#)
- link_terminated
 - auth.h, [284](#)
- LinkPhase
 - lcp.h, [300](#)
- linkoutput
 - netif, [31](#)
- lqr_period
 - lcp_options, [25](#)
- lwip_htonl
 - def.c, [51](#)
 - def.h, [160](#)
- lwip_htons
 - def.c, [52](#)
 - def.h, [160](#)
- lwip_init
 - init.c, [56](#)
 - init.h, [169](#)
- lwip_ntohl
 - def.c, [52](#)
 - def.h, [160](#)
- lwip_ntohs
 - def.c, [52](#)
 - def.h, [160](#)
- lwip_strerror
 - err.h, [166](#)
- lwip_thread_fn
 - sys.h, [261](#)
- MAX_CHALLENGE_LENGTH
 - chap.h, [288](#)
- MAX_HDR
 - vj.h, [312](#)
- MAX_NT_PASSWORD
 - chpms.h, [290](#)
- MAX_RESPONSE_LENGTH
 - chap.h, [288](#)
- MAX_SLOTS
 - vj.h, [312](#)
- MD5_CTX, [26](#)
 - buf, [27](#)
 - digest, [27](#)
 - i, [27](#)
 - in, [27](#)
- MD5_SIGNATURE_SIZE
 - chap.h, [288](#)
- MD5Final
 - md5.h, [302](#)
- MD5Init
 - md5.h, [302](#)
- MD5Update
 - md5.h, [302](#)
- MEM_ALIGNMENT
 - opt.h, [208](#)
- MEM_DEBUG
 - opt.h, [208](#)
- MEM_LIBC_MALLOC
 - opt.h, [208](#)
- MEM_SIZE
 - opt.h, [208](#)
- MEM_SIZE_ALIGNED
 - mem.c, [78](#)
- MEM_SIZE_F
 - mem.h, [171](#)
- MEM_STATS
 - opt.h, [208](#)
- MEM_STATS_AVAIL
 - stats.h, [255](#)
- MEM_STATS_DEC_USED
 - stats.h, [255](#)
- MEM_STATS_DISPLAY
 - stats.h, [255](#)
- MEM_STATS_INC
 - stats.h, [255](#)
- MEM_STATS_INC_USED
 - stats.h, [255](#)
- MEM_USE_POOLS
 - opt.h, [208](#)
- MEM_USE_POOLS_TRY_BIGGER_POOL
 - opt.h, [208](#)
- MEMCPY
 - opt.h, [208](#)
- MEMP_ALIGN_SIZE
 - memp.c, [81](#)
- MEMP_DEBUG
 - opt.h, [209](#)
- MEMP_MAX
 - memp.h, [174](#)
- MEMP_MEM_MALLOC
 - opt.h, [209](#)
- MEMP_NUM_ARP_QUEUE
 - opt.h, [209](#)
- MEMP_NUM_FRAG_PBUF
 - opt.h, [209](#)
- MEMP_NUM_IGMP_GROUP
 - opt.h, [209](#)
- MEMP_NUM_LOCALHOSTLIST
 - opt.h, [209](#)
- MEMP_NUM_NETBUF
 - opt.h, [209](#)
- MEMP_NUM_NETCONN
 - opt.h, [209](#)
- MEMP_NUM_NETDB
 - opt.h, [210](#)
- MEMP_NUM_PBUF
 - opt.h, [210](#)
- MEMP_NUM_PPPOE_INTERFACES

- opt.h, [210](#)
- MEMP_NUM_RAW_PCB
 - opt.h, [210](#)
- MEMP_NUM_REASSDATA
 - opt.h, [210](#)
- MEMP_NUM_SNMP_NODE
 - opt.h, [210](#)
- MEMP_NUM_SNMP_ROOTNODE
 - opt.h, [210](#)
- MEMP_NUM_SNMP_VALUE
 - opt.h, [210](#)
- MEMP_NUM_SNMP_VARBIND
 - opt.h, [211](#)
- MEMP_NUM_SYS_TIMEOUT
 - opt.h, [211](#)
- MEMP_NUM_TCP_PCB
 - opt.h, [211](#)
- MEMP_NUM_TCP_PCB_LISTEN
 - opt.h, [211](#)
- MEMP_NUM_TCP_SEG
 - opt.h, [211](#)
- MEMP_NUM_TCPIP_MSG_API
 - opt.h, [211](#)
- MEMP_NUM_TCPIP_MSG_INPKT
 - opt.h, [211](#)
- MEMP_NUM_UDP_PCB
 - opt.h, [211](#)
- MEMP_OVERFLOW_CHECK
 - opt.h, [212](#)
- MEMP_SANITY_CHECK
 - opt.h, [212](#)
- MEMP_SEPARATE_POOLS
 - opt.h, [212](#)
- MEMP_SIZE
 - memp.c, [81](#)
- MEMP_STATS
 - opt.h, [212](#)
- MEMP_STATS_AVAIL
 - stats.h, [256](#)
- MEMP_STATS_DEC
 - stats.h, [256](#)
- MEMP_STATS_DISPLAY
 - stats.h, [256](#)
- MEMP_STATS_INC
 - stats.h, [256](#)
- MEMP_STATS_INC_USED
 - stats.h, [256](#)
- MEMP_USE_CUSTOM_POOLS
 - opt.h, [212](#)
- MIN_CHALLENGE_LENGTH
 - chap.h, [288](#)
- MIN_SIZE
 - mem.c, [78](#)
- MIN_SIZE_ALIGNED
 - mem.c, [78](#)
- MS_CHAP_RESPONSE_LEN
 - chap.h, [288](#)
- magic
 - magic.h, [301](#)
- magic.h
 - magic, [301](#)
 - magicInit, [301](#)
- magicInit
 - magic.h, [301](#)
- magicnumber
 - lcp_options, [25](#)
- max_transmits
 - chap_state, [10](#)
- maxSlotIndex
 - vjcompress, [38](#)
- maxconfreqtransmits
 - fsm, [14](#)
- maxnakloops
 - fsm, [14](#)
- maxslotindex
 - ipcp_options, [23](#)
- maxtermtransmits
 - fsm, [14](#)
- md5.h
 - MD5Final, [302](#)
 - MD5Init, [302](#)
 - MD5Update, [302](#)
- mem, [27](#)
 - next, [28](#)
 - prev, [28](#)
 - used, [28](#)
- mem.c
 - LWIP_MEM_ALLOC_DECL_PROTECT, [77](#)
 - LWIP_MEM_ALLOC_PROTECT, [77](#)
 - LWIP_MEM_ALLOC_UNPROTECT, [77](#)
 - LWIP_MEM_FREE_DECL_PROTECT, [77](#)
 - LWIP_MEM_FREE_PROTECT, [77](#)
 - LWIP_MEM_FREE_UNPROTECT, [77](#)
 - LWIP_RAM_HEAP_POINTER, [77](#)
 - MEM_SIZE_ALIGNED, [78](#)
 - MIN_SIZE, [78](#)
 - MIN_SIZE_ALIGNED, [78](#)
 - mem_calloc, [78](#)
 - mem_free, [78](#)
 - mem_init, [78](#)
 - mem_malloc, [79](#)
 - mem_trim, [79](#)
 - ram_heap, [79](#)
 - SIZEOF_STRUCT_MEM, [78](#)
- mem.h
 - LWIP_MEM_ALIGN, [171](#)
 - LWIP_MEM_ALIGN_BUFFER, [171](#)
 - LWIP_MEM_ALIGN_SIZE, [171](#)
 - MEM_SIZE_F, [171](#)
 - mem_calloc, [171](#)
 - mem_free, [172](#)
 - mem_init, [172](#)
 - mem_malloc, [172](#)
 - mem_size_t, [171](#)
 - mem_trim, [172](#)
- mem_calloc

- mem.c, 78
- mem.h, 171
- mem_free
 - mem.c, 78
 - mem.h, 172
- mem_free_callback
 - tcpip.c, 48
 - tcpip.h, 270
- mem_init
 - mem.c, 78
 - mem.h, 172
- mem_malloc
 - mem.c, 79
 - mem.h, 172
- mem_size_t
 - mem.h, 171
- mem_trim
 - mem.c, 79
 - mem.h, 172
- memp, 28
 - next, 28
- memp.c
 - LWIP_MEMPOOL, 81
 - MEMP_ALIGN_SIZE, 81
 - MEMP_SIZE, 81
 - memp_free, 81
 - memp_init, 81
 - memp_malloc, 81
- memp.h
 - LWIP_MEMPOOL, 174
 - MEMP_MAX, 174
 - memp_free, 174
 - memp_init, 174
 - memp_malloc, 174
 - memp_t, 174
- memp_free
 - memp.c, 81
 - memp.h, 174
- memp_init
 - memp.c, 81
 - memp.h, 174
- memp_malloc
 - memp.c, 81
 - memp.h, 174
- memp_std.h
 - LWIP_MALLOC_MEMPOOL, 175
 - LWIP_MALLOC_MEMPOOL_END, 175
 - LWIP_MALLOC_MEMPOOL_START, 175
 - LWIP_PBUF_MEMPOOL, 175
- memp_t
 - memp.h, 174
- mru
 - lcp_options, 25
- msg
 - tcpip_msg, 36
- mtu
 - netif, 31
- NETBUF_FLAG_CHKSUM
 - netbuf.h, 177
- NETBUF_FLAG_DESTADDR
 - netbuf.h, 177
- NETIF_DEBUG
 - opt.h, 212
- NETIF_FLAG_BROADCAST
 - netif.h, 182
- NETIF_FLAG_DHCP
 - netif.h, 183
- NETIF_FLAG_ETHARP
 - netif.h, 183
- NETIF_FLAG_ETHERNET
 - netif.h, 183
- NETIF_FLAG_IGMP
 - netif.h, 183
- NETIF_FLAG_LINK_UP
 - netif.h, 183
- NETIF_FLAG_POINTTOPOINT
 - netif.h, 183
- NETIF_FLAG_UP
 - netif.h, 183
- NETIF_INIT_SNMP
 - netif.h, 183
- NETIF_LINK_CALLBACK
 - netif.c, 83
- NETIF_MAX_HWADDR_LEN
 - netif.h, 184
- NETIF_SET_HWADDRHINT
 - netif.h, 184
- NETIF_STATUS_CALLBACK
 - netif.c, 83
- NEW_A
 - vj.h, 312
- NEW_C
 - vj.h, 312
- NEW_I
 - vj.h, 312
- NEW_S
 - vj.h, 312
- NEW_U
 - vj.h, 312
- NEW_W
 - vj.h, 313
- NO_SYS
 - opt.h, 212
- NO_SYS_NO_TIMERS
 - opt.h, 212
- NULL
 - def.h, 161
- nakci
 - fsm_callbacks, 16
- nakloops
 - fsm, 14
- name
 - netif, 31
- neg_accompression
 - lcp_options, 25
- neg_addr

- ipcp_options, 23
- neg_asyncmap
 - lcp_options, 25
- neg_cbc
 - lcp_options, 25
- neg_chap
 - lcp_options, 25
- neg_lqr
 - lcp_options, 26
- neg_magicnumber
 - lcp_options, 26
- neg_mru
 - lcp_options, 26
- neg_pcompression
 - lcp_options, 26
- neg_upap
 - lcp_options, 26
- neg_vj
 - ipcp_options, 23
- netbuf, 29
 - addr, 29
 - p, 29
 - port, 29
 - ptr, 29
- netbuf.h
 - NETBUF_FLAG_CHKSUM, 177
 - NETBUF_FLAG_DESTADDR, 177
 - netbuf_alloc, 178
 - netbuf_chain, 178
 - netbuf_copy, 177
 - netbuf_copy_partial, 177
 - netbuf_data, 178
 - netbuf_delete, 178
 - netbuf_first, 178
 - netbuf_free, 178
 - netbuf_fromaddr, 177
 - netbuf_fromport, 177
 - netbuf_len, 177
 - netbuf_new, 178
 - netbuf_next, 178
 - netbuf_ref, 178
 - netbuf_set_fromaddr, 177
 - netbuf_take, 178
- netbuf_alloc
 - netbuf.h, 178
- netbuf_chain
 - netbuf.h, 178
- netbuf_copy
 - netbuf.h, 177
- netbuf_copy_partial
 - netbuf.h, 177
- netbuf_data
 - netbuf.h, 178
- netbuf_delete
 - netbuf.h, 178
- netbuf_first
 - netbuf.h, 178
- netbuf_free
 - netbuf.h, 178
- netbuf_fromaddr
 - netbuf.h, 177
- netbuf_fromport
 - netbuf.h, 177
- netbuf_len
 - netbuf.h, 177
- netbuf_new
 - netbuf.h, 178
- netbuf_next
 - netbuf.h, 178
- netbuf_ref
 - netbuf.h, 178
- netbuf_set_fromaddr
 - netbuf.h, 177
- netbuf_take
 - netbuf.h, 178
- netif, 30
 - flags, 30
 - gw, 30
 - hwaddr, 31
 - hwaddr_len, 31
 - input, 31
 - ip_addr, 31
 - linkoutput, 31
 - mtu, 31
 - name, 31
 - netmask, 31
 - next, 31
 - num, 31
 - output, 32
 - state, 32
 - tcpip_msg, 36
- netif.c
 - NETIF_LINK_CALLBACK, 83
 - NETIF_STATUS_CALLBACK, 83
 - netif_add, 83
 - netif_default, 87
 - netif_find, 83
 - netif_init, 84
 - netif_list, 87
 - netif_remove, 84
 - netif_set_addr, 84
 - netif_set_default, 84
 - netif_set_down, 84
 - netif_set_gw, 84
 - netif_set_ipaddr, 86
 - netif_set_link_down, 86
 - netif_set_link_up, 86
 - netif_set_netmask, 86
 - netif_set_up, 86
- netif.h
 - ENABLE_LOOPBACK, 182
 - NETIF_FLAG_BROADCAST, 182
 - NETIF_FLAG_DHCP, 183
 - NETIF_FLAG_ETHARP, 183
 - NETIF_FLAG_ETHERNET, 183
 - NETIF_FLAG_IGMP, 183

- NETIF_FLAG_LINK_UP, 183
- NETIF_FLAG_POINTTOPOINT, 183
- NETIF_FLAG_UP, 183
- NETIF_INIT_SNMP, 183
- NETIF_MAX_HWADDR_LEN, 184
- NETIF_SET_HWADDRHINT, 184
- netif_add, 185
- netif_default, 189
- netif_find, 185
- netif_igmp_mac_filter_fn, 184
- netif_init, 186
- netif_init_fn, 184
- netif_input_fn, 184
- netif_is_link_up, 183
- netif_is_up, 184
- netif_linkoutput_fn, 184
- netif_list, 189
- netif_output_fn, 185
- netif_remove, 186
- netif_set_addr, 186
- netif_set_default, 186
- netif_set_down, 186
- netif_set_gw, 186
- netif_set_ipaddr, 188
- netif_set_link_down, 188
- netif_set_link_up, 188
- netif_set_netmask, 188
- netif_set_up, 188
- netif_status_callback_fn, 185
- netif_add
 - netif.c, 83
 - netif.h, 185
- netif_default
 - netif.c, 87
 - netif.h, 189
- netif_find
 - netif.c, 83
 - netif.h, 185
- netif_igmp_mac_filter_fn
 - netif.h, 184
- netif_init
 - netif.c, 84
 - netif.h, 186
- netif_init_fn
 - netif.h, 184
- netif_input_fn
 - netif.h, 184
- netif_is_link_up
 - netif.h, 183
- netif_is_up
 - netif.h, 184
- netif_linkoutput_fn
 - netif.h, 184
- netif_list
 - netif.c, 87
 - netif.h, 189
- netif_output_fn
 - netif.h, 185
- netif_remove
 - netif.c, 84
 - netif.h, 186
- netif_set_addr
 - netif.c, 84
 - netif.h, 186
- netif_set_default
 - netif.c, 84
 - netif.h, 186
- netif_set_down
 - netif.c, 84
 - netif.h, 186
- netif_set_gw
 - netif.c, 84
 - netif.h, 186
- netif_set_ipaddr
 - netif.c, 86
 - netif.h, 188
- netif_set_link_down
 - netif.c, 86
 - netif.h, 188
- netif_set_link_up
 - netif.c, 86
 - netif.h, 188
- netif_set_netmask
 - netif.c, 86
 - netif.h, 188
- netif_set_up
 - netif.c, 86
 - netif.h, 188
- netif_status_callback_fn
 - netif.h, 185
- netmask
 - netif, 31
- next
 - mem, 28
 - memp, 28
 - netif, 31
 - pbuf, 33
 - sys_timeo, 34
- nextthr
 - ip_hdr, 21
- np_down
 - auth.h, 284
- np_finished
 - auth.h, 284
- np_up
 - auth.h, 284
- ntohl
 - def.h, 161
 - ipv6/lwip/inet.h, 122
- ntohs
 - def.h, 161
 - ipv6/lwip/inet.h, 122
- num
 - netif, 31
- numloops
 - lcp_options, 26

- OPT_PASSIVE
 - fsm.h, [293](#)
- OPT_RESTART
 - fsm.h, [293](#)
- OPT_SILENT
 - fsm.h, [293](#)
- old_addrs
 - ipcp_options, [23](#)
- old_vj
 - ipcp_options, [23](#)
- opt.h
 - API_LIB_DEBUG, [195](#)
 - API_MSG_DEBUG, [195](#)
 - ARP_QUEUEING, [195](#)
 - ARP_TABLE_SIZE, [195](#)
 - AUTOIP_DEBUG, [195](#)
 - CHECKSUM_CHECK_IP, [195](#)
 - CHECKSUM_CHECK_TCP, [195](#)
 - CHECKSUM_CHECK_UDP, [195](#)
 - CHECKSUM_GEN_ICMP, [195](#)
 - CHECKSUM_GEN_IP, [196](#)
 - CHECKSUM_GEN_TCP, [196](#)
 - CHECKSUM_GEN_UDP, [196](#)
 - DEFAULT_ACCEPTMBOX_SIZE, [196](#)
 - DEFAULT_RAW_RECVMBOX_SIZE, [196](#)
 - DEFAULT_TCP_RECVMBOX_SIZE, [196](#)
 - DEFAULT_THREAD_NAME, [196](#)
 - DEFAULT_THREAD_PRIO, [196](#)
 - DEFAULT_THREAD_STACKSIZE, [196](#)
 - DEFAULT_UDP_RECVMBOX_SIZE, [197](#)
 - DHCP_DEBUG, [197](#)
 - DHCP_DOES_ARP_CHECK, [197](#)
 - DNS_DEBUG, [197](#)
 - DNS_DOES_NAME_CHECK, [197](#)
 - DNS_LOCAL_HOSTLIST, [197](#)
 - DNS_LOCAL_HOSTLIST_IS_DYNAMIC, [197](#)
 - DNS_MAX_NAME_LENGTH, [197](#)
 - DNS_MAX_SERVERS, [198](#)
 - DNS_MSG_SIZE, [198](#)
 - DNS_TABLE_SIZE, [198](#)
 - ETH_PAD_SIZE, [198](#)
 - ETHARP_DEBUG, [198](#)
 - ETHARP_STATS, [198](#)
 - ETHARP_SUPPORT_STATIC_ENTRIES, [198](#)
 - ETHARP_SUPPORT_VLAN, [198](#)
 - ETHARP_TRUST_IP_MAC, [199](#)
 - ICMP_DEBUG, [199](#)
 - ICMP_STATS, [199](#)
 - ICMP_TTL, [199](#)
 - IGMP_DEBUG, [199](#)
 - IGMP_STATS, [199](#)
 - INET_DEBUG, [199](#)
 - IP_DEBUG, [199](#)
 - IP_DEFAULT_TTL, [199](#)
 - IP_FORWARD, [200](#)
 - IP_FORWARD_ALLOW_TX_ON_RX_NETIF, [200](#)
 - IP_FRAG, [200](#)
 - IP_FRAG_USES_STATIC_BUF, [200](#)
 - IP_OPTIONS_ALLOWED, [200](#)
 - IP_REASS_DEBUG, [200](#)
 - IP_REASS_MAX_PBUFS, [200](#)
 - IP_REASS_MAXAGE, [201](#)
 - IP_REASSEMBLY, [201](#)
 - IP_SOF_BROADCAST, [201](#)
 - IP_SOF_BROADCAST_RECV, [201](#)
 - IP_STATS, [201](#)
 - IPFRAG_STATS, [201](#)
 - LINK_STATS, [201](#)
 - LWIP_ALLOW_MEM_FREE_FROM_OTHER_CONTEXT, [201](#)
 - LWIP_ARP, [202](#)
 - LWIP_AUTOIP, [202](#)
 - LWIP_BROADCAST_PING, [202](#)
 - LWIP_CALLBACK_API, [202](#)
 - LWIP_CHECKSUM_ON_COPY, [202](#)
 - LWIP_COMPAT_SOCKETS, [202](#)
 - LWIP_DBG_MIN_LEVEL, [202](#)
 - LWIP_DBG_TYPES_ON, [203](#)
 - LWIP_DHCP, [203](#)
 - LWIP_DHCP_AUTOIP_COOP, [203](#)
 - LWIP_DHCP_AUTOIP_COOP_TRIES, [203](#)
 - LWIP_DNS, [203](#)
 - LWIP_ETHERNET, [203](#)
 - LWIP_EVENT_API, [203](#)
 - LWIP_HAVE_LOOPIF, [203](#)
 - LWIP_HAVE_SLIPIF, [204](#)
 - LWIP_ICMP, [204](#)
 - LWIP_IGMP, [204](#)
 - LWIP_LOOPBACK_MAX_PBUFS, [204](#)
 - LWIP_MULTICAST_PING, [204](#)
 - LWIP_NETBUF_RECVINFO, [204](#)
 - LWIP_NETCONN, [204](#)
 - LWIP_NETIF_API, [204](#)
 - LWIP_NETIF_HOSTNAME, [204](#)
 - LWIP_NETIF_HWADDRHINT, [205](#)
 - LWIP_NETIF_LINK_CALLBACK, [205](#)
 - LWIP_NETIF_LOOPBACK, [205](#)
 - LWIP_NETIF_LOOPBACK_MULTITHREADING, [205](#)
 - LWIP_NETIF_REMOVE_CALLBACK, [205](#)
 - LWIP_NETIF_STATUS_CALLBACK, [205](#)
 - LWIP_NETIF_TX_SINGLE_PBUF, [205](#)
 - LWIP_POSIX_SOCKETS_IO_NAMES, [206](#)
 - LWIP_RANDOMIZE_INITIAL_LOCAL_PORTS, [206](#)
 - LWIP_RAW, [206](#)
 - LWIP_SNMP, [206](#)
 - LWIP_SO_RCVBUF, [206](#)
 - LWIP_SO_RCVTIMEO, [206](#)
 - LWIP_SO_SNDTIMEO, [206](#)
 - LWIP_SOCKET, [206](#)
 - LWIP_STATS, [206](#)
 - LWIP_STATS_DISPLAY, [207](#)
 - LWIP_TCP, [207](#)
 - LWIP_TCP_KEEPALIVE, [207](#)
 - LWIP_TCP_TIMESTAMPS, [207](#)

LWIP_TCPIP_CORE_LOCKING, 207
LWIP_TCPIP_CORE_LOCKING_INPUT, 207
LWIP_TCPIP_TIMEOUT, 207
LWIP_UDP, 207
LWIP_UDPLITE, 207
MEM_ALIGNMENT, 208
MEM_DEBUG, 208
MEM_LIBC_MALLOC, 208
MEM_SIZE, 208
MEM_STATS, 208
MEM_USE_POOLS, 208
MEM_USE_POOLS_TRY_BIGGER_POOL, 208
MEMCPY, 208
MEMP_DEBUG, 209
MEMP_MEM_MALLOC, 209
MEMP_NUM_ARP_QUEUE, 209
MEMP_NUM_FRAG_PBUF, 209
MEMP_NUM_IGMP_GROUP, 209
MEMP_NUM_LOCALHOSTLIST, 209
MEMP_NUM_NETBUF, 209
MEMP_NUM_NETCONN, 209
MEMP_NUM_NETDB, 210
MEMP_NUM_PBUF, 210
MEMP_NUM_PPPOE_INTERFACES, 210
MEMP_NUM_RAW_PCB, 210
MEMP_NUM_REASSDATA, 210
MEMP_NUM_SNMP_NODE, 210
MEMP_NUM_SNMP_ROOTNODE, 210
MEMP_NUM_SNMP_VALUE, 210
MEMP_NUM_SNMP_VARBIND, 211
MEMP_NUM_SYS_TIMEOUT, 211
MEMP_NUM_TCP_PCB, 211
MEMP_NUM_TCP_PCB_LISTEN, 211
MEMP_NUM_TCP_SEG, 211
MEMP_NUM_TCPIP_MSG_API, 211
MEMP_NUM_TCPIP_MSG_INPKT, 211
MEMP_NUM_UDP_PCB, 211
MEMP_OVERFLOW_CHECK, 212
MEMP_SANITY_CHECK, 212
MEMP_SEPARATE_POOLS, 212
MEMP_STATS, 212
MEMP_USE_CUSTOM_POOLS, 212
NETIF_DEBUG, 212
NO_SYS, 212
NO_SYS_NO_TIMERS, 212
PBUF_DEBUG, 213
PBUF_LINK_HLEN, 213
PBUF_POOL_BUFSIZE, 213
PBUF_POOL_SIZE, 213
PPP_DEBUG, 213
PPP_SUPPORT, 213
PPP_THREAD_NAME, 213
PPP_THREAD_PRIO, 213
PPP_THREAD_STACKSIZE, 213
PPPOE_SUPPORT, 214
PPPOS_SUPPORT, 214
RAW_DEBUG, 214
RAW_TTL, 214
RCV_BUFSIZE_DEFAULT, 214
SLIP_DEBUG, 214
SLIIF_THREAD_NAME, 214
SLIIF_THREAD_PRIO, 214
SLIIF_THREAD_STACKSIZE, 214
SMEMCPY, 215
SNMP_CONCURRENT_REQUESTS, 215
SNMP_MAX_OCTET_STRING_LEN, 215
SNMP_MAX_TREE_DEPTH, 215
SNMP_MAX_VALUE_SIZE, 215
SNMP_MIB_DEBUG, 215
SNMP_MSG_DEBUG, 215
SNMP_PRIVATE_MIB, 215
SNMP_SAFE_REQUESTS, 216
SNMP_TRAP_DESTINATIONS, 216
SO_REUSE, 216
SO_REUSE_RXTOALL, 216
SOCKETS_DEBUG, 216
SYS_DEBUG, 216
SYS_LIGHTWEIGHT_PROT, 216
SYS_STATS, 216
TCP_CALCULATE_EFF_SEND_MSS, 216
TCP_CWND_DEBUG, 217
TCP_DEBUG, 217
TCP_DEFAULT_LISTEN_BACKLOG, 217
TCP_FR_DEBUG, 217
TCP_INPUT_DEBUG, 217
TCP_LISTEN_BACKLOG, 217
TCP_MAXRTX, 217
TCP_MSS, 217
TCP_OOSEQ_MAX_BYTES, 218
TCP_OOSEQ_MAX_PBUFS, 218
TCP_OUTPUT_DEBUG, 218
TCP_OVERSIZE, 218
TCP_QLEN_DEBUG, 218
TCP_QUEUE_OOSEQ, 218
TCP_RST_DEBUG, 218
TCP_RTO_DEBUG, 218
TCP_SND_BUF, 219
TCP_SND_QUEUELEN, 219
TCP_SNDLOWAT, 219
TCP_SNDQUEUELOWAT, 219
TCP_STATS, 219
TCP_SYNMAXRTX, 219
TCP_TTL, 219
TCP_WND, 219
TCP_WND_DEBUG, 219
TCP_WND_UPDATE_THRESHOLD, 220
TCPIP_DEBUG, 220
TCPIP_MBOX_SIZE, 220
TCPIP_THREAD_NAME, 220
TCPIP_THREAD_PRIO, 220
TCPIP_THREAD_STACKSIZE, 220
TIMERS_DEBUG, 220
UDP_DEBUG, 220
UDP_STATS, 220
UDP_TTL, 221
ouraddr

- ipcp_options, [24](#)
- output
 - netif, [32](#)
- p
 - netbuf, [29](#)
 - tcpip_msg, [36](#)
- PACK_STRUCT_BEGIN
 - arch.h, [155](#)
- PACK_STRUCT_END
 - arch.h, [156](#)
- PACK_STRUCT_FIELD
 - arch.h, [156](#)
 - icmp_echo_hdr, [17](#)
 - ip_addr, [18](#)
 - ip_addr2, [19](#)
 - ip_addr_packed, [19](#)
 - ip_hdr, [20, 21](#)
- PACK_STRUCT_STRUCT
 - ipv4/lwip/icmp.h, [113](#)
 - ipv4/lwip/ip.h, [134](#)
 - ipv4/lwip/ip_addr.h, [148](#)
 - ipv6/lwip/ip_addr.h, [151](#)
- PBUF_DEBUG
 - opt.h, [213](#)
- PBUF_FLAG_IS_CUSTOM
 - pbuf.h, [223](#)
- PBUF_FLAG_LLBCAST
 - pbuf.h, [223](#)
- PBUF_FLAG_LLMCAST
 - pbuf.h, [223](#)
- PBUF_FLAG_MCASTLOOP
 - pbuf.h, [223](#)
- PBUF_FLAG_PUSH
 - pbuf.h, [223](#)
- PBUF_FLAG_TCP_FIN
 - pbuf.h, [223](#)
- PBUF_IP
 - pbuf.h, [224](#)
- PBUF_IP_HLEN
 - pbuf.h, [223](#)
- PBUF_LINK
 - pbuf.h, [224](#)
- PBUF_LINK_HLEN
 - opt.h, [213](#)
- PBUF_POOL
 - pbuf.h, [224](#)
- PBUF_POOL_BUFSIZE
 - opt.h, [213](#)
- PBUF_POOL_BUFSIZE_ALIGNED
 - pbuf.c, [88](#)
- PBUF_POOL_IS_EMPTY
 - pbuf.c, [88](#)
- PBUF_POOL_SIZE
 - opt.h, [213](#)
- PBUF_RAM
 - pbuf.h, [224](#)
- PBUF_RAW
 - pbuf.h, [224](#)
- PBUF_REF
 - pbuf.h, [224](#)
- PBUF_ROM
 - pbuf.h, [224](#)
- PBUF_TRANSPORT
 - pbuf.h, [224](#)
- PBUF_TRANSPORT_HLEN
 - pbuf.h, [223](#)
- PHASE_AUTHENTICATE
 - lcp.h, [300](#)
- PHASE_CALLBACK
 - lcp.h, [300](#)
- PHASE_DEAD
 - lcp.h, [300](#)
- PHASE_ESTABLISH
 - lcp.h, [300](#)
- PHASE_INITIALIZE
 - lcp.h, [300](#)
- PHASE_NETWORK
 - lcp.h, [300](#)
- PHASE_TERMINATE
 - lcp.h, [300](#)
- PP_HTONL
 - def.h, [161](#)
- PP_HTONS
 - def.h, [161](#)
- PP_NTOHL
 - def.h, [161](#)
- PP_NTOHS
 - def.h, [161](#)
- PPP_DEBUG
 - opt.h, [213](#)
- PPP_SUPPORT
 - opt.h, [213](#)
- PPP_THREAD_NAME
 - opt.h, [213](#)
- PPP_THREAD_PRIO
 - opt.h, [213](#)
- PPP_THREAD_STACKSIZE
 - opt.h, [213](#)
- PPPDEBUG
 - pppdebug.h, [308](#)
- PPPOE_SUPPORT
 - opt.h, [214](#)
- PPPOS_SUPPORT
 - opt.h, [214](#)
- PROTREJ
 - lcp.h, [300](#)
- passive
 - lcp_options, [26](#)
- payload
 - pbuf, [33](#)
- pbuf, [32](#)
 - flags, [33](#)
 - len, [33](#)
 - next, [33](#)
 - payload, [33](#)
 - ref, [33](#)

- tot_len, [33](#)
- type, [33](#)
- pbuf.c
 - PBUF_POOL_BUFSIZE_ALIGNED, [88](#)
 - PBUF_POOL_IS_EMPTY, [88](#)
 - pbuf_alloc, [89](#)
 - pbuf_cat, [89](#)
 - pbuf_chain, [89](#)
 - pbuf_clen, [90](#)
 - pbuf_coalesce, [90](#)
 - pbuf_copy, [90](#)
 - pbuf_copy_partial, [91](#)
 - pbuf_dechain, [91](#)
 - pbuf_free, [91](#)
 - pbuf_get_at, [92](#)
 - pbuf_header, [92](#)
 - pbuf_memcmp, [92](#)
 - pbuf_memfind, [93](#)
 - pbuf_realloc, [93](#)
 - pbuf_ref, [93](#)
 - pbuf_strstr, [94](#)
 - pbuf_take, [94](#)
 - SIZEOF_STRUCT_PBUF, [88](#)
- pbuf.h
 - LWIP_SUPPORT_CUSTOM_PBUF, [223](#)
 - PBUF_FLAG_IS_CUSTOM, [223](#)
 - PBUF_FLAG_LLBCAST, [223](#)
 - PBUF_FLAG_LLMCAST, [223](#)
 - PBUF_FLAG_MCASTLOOP, [223](#)
 - PBUF_FLAG_PUSH, [223](#)
 - PBUF_FLAG_TCP_FIN, [223](#)
 - PBUF_IP, [224](#)
 - PBUF_IP_HLEN, [223](#)
 - PBUF_LINK, [224](#)
 - PBUF_POOL, [224](#)
 - PBUF_RAM, [224](#)
 - PBUF_RAW, [224](#)
 - PBUF_REF, [224](#)
 - PBUF_ROM, [224](#)
 - PBUF_TRANSPORT, [224](#)
 - PBUF_TRANSPORT_HLEN, [223](#)
 - pbuf_alloc, [224](#)
 - pbuf_cat, [225](#)
 - pbuf_chain, [225](#)
 - pbuf_clen, [225](#)
 - pbuf_coalesce, [225](#)
 - pbuf_copy, [226](#)
 - pbuf_copy_partial, [226](#)
 - pbuf_dechain, [226](#)
 - pbuf_free, [227](#)
 - pbuf_get_at, [227](#)
 - pbuf_header, [227](#)
 - pbuf_init, [223](#)
 - pbuf_layer, [224](#)
 - pbuf_memcmp, [228](#)
 - pbuf_memfind, [228](#)
 - pbuf_realloc, [228](#)
 - pbuf_ref, [229](#)
 - pbuf_strstr, [229](#)
 - pbuf_take, [229](#)
 - pbuf_type, [224](#)
- pbuf_alloc
 - pbuf.c, [89](#)
 - pbuf.h, [224](#)
- pbuf_cat
 - pbuf.c, [89](#)
 - pbuf.h, [225](#)
- pbuf_chain
 - pbuf.c, [89](#)
 - pbuf.h, [225](#)
- pbuf_clen
 - pbuf.c, [90](#)
 - pbuf.h, [225](#)
- pbuf_coalesce
 - pbuf.c, [90](#)
 - pbuf.h, [225](#)
- pbuf_copy
 - pbuf.c, [90](#)
 - pbuf.h, [226](#)
- pbuf_copy_partial
 - pbuf.c, [91](#)
 - pbuf.h, [226](#)
- pbuf_dechain
 - pbuf.c, [91](#)
 - pbuf.h, [226](#)
- pbuf_free
 - pbuf.c, [91](#)
 - pbuf.h, [227](#)
- pbuf_free_callback
 - tcpip.c, [48](#)
 - tcpip.h, [270](#)
- pbuf_get_at
 - pbuf.c, [92](#)
 - pbuf.h, [227](#)
- pbuf_header
 - pbuf.c, [92](#)
 - pbuf.h, [227](#)
- pbuf_init
 - pbuf.h, [223](#)
- pbuf_layer
 - pbuf.h, [224](#)
- pbuf_memcmp
 - pbuf.c, [92](#)
 - pbuf.h, [228](#)
- pbuf_memfind
 - pbuf.c, [93](#)
 - pbuf.h, [228](#)
- pbuf_realloc
 - pbuf.c, [93](#)
 - pbuf.h, [228](#)
- pbuf_ref
 - pbuf.c, [93](#)
 - pbuf.h, [229](#)
- pbuf_strstr
 - pbuf.c, [94](#)
 - pbuf.h, [229](#)

- pbuf_take
 - pbuf.c, [94](#)
 - pbuf.h, [229](#)
- pbuf_type
 - pbuf.h, [224](#)
- peer_mru
 - fsm.h, [294](#)
- port
 - netbuf, [29](#)
- pppdebug.h
 - AUTHDEBUG, [307](#)
 - CHAPDEBUG, [307](#)
 - FSMDEBUG, [308](#)
 - IPCPDEBUG, [308](#)
 - LCPDEBUG, [308](#)
 - LOG_CRITICAL, [308](#)
 - LOG_DEBUG, [308](#)
 - LOG_DETAIL, [308](#)
 - LOG_ERR, [308](#)
 - LOG_INFO, [308](#)
 - LOG_NOTICE, [308](#)
 - LOG_WARNING, [308](#)
 - PPPDEBUG, [308](#)
 - TRACELCP, [308](#)
 - UPAPDEBUG, [309](#)
- prev
 - mem, [28](#)
- proto_name
 - fsm_callbacks, [16](#)
- protocol
 - fsm, [14](#)
- protreject
 - fsm_callbacks, [16](#)
- proxy_arp
 - ipcp_options, [24](#)
- ptr
 - netbuf, [29](#)
- RAW_DEBUG
 - opt.h, [214](#)
- RAW_TTL
 - opt.h, [214](#)
- RECV_BUFSIZE_DEFAULT
 - opt.h, [214](#)
- ram_heap
 - mem.c, [79](#)
- randm.h
 - avChurnRand, [310](#)
 - avGenRand, [310](#)
 - avRandom, [310](#)
 - avRandomInit, [310](#)
 - avRandomize, [310](#)
- ref
 - pbuf, [33](#)
- rejci
 - fsm_callbacks, [16](#)
- req_addr
 - ipcp_options, [24](#)
- req_dns1
 - ipcp_options, [24](#)
- req_dns2
 - ipcp_options, [24](#)
- reqci
 - fsm_callbacks, [16](#)
- reqid
 - fsm, [14](#)
- resetci
 - fsm_callbacks, [16](#)
- resp_id
 - chap_state, [10](#)
- resp_length
 - chap_state, [10](#)
- resp_name
 - chap_state, [10](#)
- resp_transmits
 - chap_state, [10](#)
- resp_type
 - chap_state, [11](#)
- response
 - chap_state, [11](#)
- restart
 - lcp_options, [26](#)
- retransmit
 - fsm_callbacks, [16](#)
- retransmits
 - fsm, [14](#)
- rstate
 - vjcompress, [38](#)
- s_addr
 - in_addr, [18](#)
- sizeof_struct_mem
 - mem.c, [78](#)
- sizeof_struct_pbuf
 - pbuf.c, [88](#)
- SLIP_DEBUG
 - opt.h, [214](#)
- SLIP_RX_FROM_ISR
 - slipif.h, [279](#)
- SLIP_RX_QUEUE
 - slipif.h, [279](#)
- SLIP_USE_RX_THREAD
 - slipif.h, [279](#)
- SLIPIF_THREAD_NAME
 - opt.h, [214](#)
- SLIPIF_THREAD_PRIO
 - opt.h, [214](#)
- SLIPIF_THREAD_STACKSIZE
 - opt.h, [214](#)
- SMEMCPY
 - opt.h, [215](#)
- SNMP_CONCURRENT_REQUESTS
 - opt.h, [215](#)
- SNMP_MAX_OCTET_STRING_LEN
 - opt.h, [215](#)
- SNMP_MAX_TREE_DEPTH
 - opt.h, [215](#)
- SNMP_MAX_VALUE_SIZE

- opt.h, 215
- SNMP_MIB_DEBUG
 - opt.h, 215
- SNMP_MSG_DEBUG
 - opt.h, 215
- SNMP_PRIVATE_MIB
 - opt.h, 215
- SNMP_SAFE_REQUESTS
 - opt.h, 216
- SNMP_TRAP_DESTINATIONS
 - opt.h, 216
- SO_REUSE
 - opt.h, 216
- SO_REUSE_RXTOALL
 - opt.h, 216
- SOCKETS_DEBUG
 - opt.h, 216
- SOF_ACCEPTCONN
 - ipv4/lwip/ip.h, 131
- SOF_BROADCAST
 - ipv4/lwip/ip.h, 131
- SOF_INHERITED
 - ipv4/lwip/ip.h, 131
- SOF_KEEPALIVE
 - ipv4/lwip/ip.h, 131
- SOF_LINGER
 - ipv4/lwip/ip.h, 131
- SOF_REUSEADDR
 - ipv4/lwip/ip.h, 131
- SPECIAL_D
 - vj.h, 313
- SPECIAL_I
 - vj.h, 313
- SPECIALS_MASK
 - vj.h, 313
- STATS_DEC
 - stats.h, 256
- STATS_INC
 - stats.h, 257
- STATS_INC_USED
 - stats.h, 257
- SWAP_BYTES_IN_WORD
 - inet_chksum.h, 124
- SYS_ARCH_DEC
 - sys.h, 259
- SYS_ARCH_DECL_PROTECT
 - sys.h, 260
- SYS_ARCH_GET
 - sys.h, 260
- SYS_ARCH_INC
 - sys.h, 260
- SYS_ARCH_PROTECT
 - sys.h, 260
- SYS_ARCH_SET
 - sys.h, 260
- SYS_ARCH_TIMEOUT
 - sys.h, 260
- SYS_ARCH_UNPROTECT
 - sys.h, 261
- SYS_DEBUG
 - opt.h, 216
- SYS_LIGHTWEIGHT_PROT
 - opt.h, 216
- SYS_MBOX_EMPTY
 - sys.h, 261
- SYS_STATS
 - opt.h, 216
- SYS_STATS_DEC
 - stats.h, 257
- SYS_STATS_DISPLAY
 - stats.h, 257
- SYS_STATS_INC
 - stats.h, 257
- SYS_STATS_INC_USED
 - stats.h, 257
- SZT_F
 - arch.h, 156
- seen_ack
 - fsm, 14
- sem
 - tcpip_msg, 36
- serverstate
 - chap_state, 11
- silent
 - lcp_options, 26
- sio.h
 - sio_fd_t, 231
 - sio_open, 231
 - sio_read, 231
 - sio_read_abort, 233
 - sio_recv, 233
 - sio_send, 233
 - sio_tryread, 233
 - sio_write, 234
- sio_fd_t
 - sio.h, 231
- sio_open
 - sio.h, 231
- sio_read
 - sio.h, 231
- sio_read_abort
 - sio.h, 233
- sio_recv
 - sio.h, 233
- sio_send
 - sio.h, 233
- sio_tryread
 - sio.h, 233
- sio_write
 - sio.h, 234
- slipif.h
 - SLIP_RX_FROM_ISR, 279
 - SLIP_RX_QUEUE, 279
 - SLIP_USE_RX_THREAD, 279
 - slipif_init, 279
 - slipif_poll, 279

- slipif_init
 - slipif.h, 279
- slipif_poll
 - slipif.h, 279
- snmp.h
 - snmp_add_ifinOctets, 238
 - snmp_add_ifoutOctets, 238
 - snmp_add_snmpinttotalreqvars, 238
 - snmp_add_snmpinttotalsetvars, 238
 - snmp_add_sysuptime, 238
 - snmp_dec_iflist, 238
 - snmp_delete_arpidx_tree, 238
 - snmp_delete_ipaddridx_tree, 238
 - snmp_delete_iprtidx_tree, 238
 - snmp_delete_udpidx_tree, 238
 - snmp_get_snmpenableauthentraps, 239
 - snmp_get_snmpgrpid_ptr, 239
 - snmp_get_sysobjid_ptr, 239
 - snmp_get_sysuptime, 239
 - snmp_ifType, 247
 - snmp_ifType_basicISDN, 248
 - snmp_ifType_ddn_x25, 247
 - snmp_ifType_ds1, 248
 - snmp_ifType_ds3, 248
 - snmp_ifType_e1, 248
 - snmp_ifType_eon, 248
 - snmp_ifType_ethernet_3Mbit, 248
 - snmp_ifType_ethernet_csmacd, 247
 - snmp_ifType_fddi, 248
 - snmp_ifType_frame_relay, 248
 - snmp_ifType_hdh1822, 247
 - snmp_ifType_hyperchannel, 248
 - snmp_ifType_iso88023_csmacd, 247
 - snmp_ifType_iso88024_tokenBus, 247
 - snmp_ifType_iso88025_tokenRing, 247
 - snmp_ifType_iso88026_man, 248
 - snmp_ifType_lapb, 248
 - snmp_ifType_nsip, 248
 - snmp_ifType_other, 247
 - snmp_ifType_ppp, 248
 - snmp_ifType_primaryISDN, 248
 - snmp_ifType_propPointToPointSerial, 248
 - snmp_ifType_proteon_10Mbit, 248
 - snmp_ifType_proteon_80Mbit, 248
 - snmp_ifType_regular1822, 247
 - snmp_ifType_rfc877_x25, 247
 - snmp_ifType_sdlc, 248
 - snmp_ifType_sip, 248
 - snmp_ifType_slip, 248
 - snmp_ifType_softwareLoopback, 248
 - snmp_ifType_starLan, 248
 - snmp_ifType_ultra, 248
 - snmp_inc_icmpinaddrmaskreps, 239
 - snmp_inc_icmpinaddrmasks, 239
 - snmp_inc_icmpindestunreaches, 239
 - snmp_inc_icmpinechoreps, 239
 - snmp_inc_icmpinechos, 239
 - snmp_inc_icmpinerrors, 239
 - snmp_inc_icmpinmsgs, 239
 - snmp_inc_icmpinparmprobs, 239
 - snmp_inc_icmpinredirects, 240
 - snmp_inc_icmpinsrcquenches, 240
 - snmp_inc_icmpintimeexcds, 240
 - snmp_inc_icmpintimestamprobs, 240
 - snmp_inc_icmpintimestamps, 240
 - snmp_inc_icmpoutaddrmaskreps, 240
 - snmp_inc_icmpoutaddrmasks, 240
 - snmp_inc_icmpoutdestunreaches, 240
 - snmp_inc_icmpoutechoreps, 240
 - snmp_inc_icmpoutechos, 240
 - snmp_inc_icmpouterrors, 240
 - snmp_inc_icmpoutmsgs, 240
 - snmp_inc_icmpoutparmprobs, 241
 - snmp_inc_icmpoutredirects, 241
 - snmp_inc_icmpoutsrcquenches, 241
 - snmp_inc_icmpouttimeexcds, 241
 - snmp_inc_icmpouttimestamprobs, 241
 - snmp_inc_icmpouttimestamps, 241
 - snmp_inc_ifindiscards, 241
 - snmp_inc_ifinnucastpkts, 241
 - snmp_inc_ifinucastpkts, 241
 - snmp_inc_iflist, 241
 - snmp_inc_ifoutdiscards, 241
 - snmp_inc_ifoutnucastpkts, 241
 - snmp_inc_ifoutucastpkts, 242
 - snmp_inc_ipforwdatagrams, 242
 - snmp_inc_ipfragcreates, 242
 - snmp_inc_ipfragfails, 242
 - snmp_inc_ipfragoks, 242
 - snmp_inc_ipinaddrerrors, 242
 - snmp_inc_ipindeliivers, 242
 - snmp_inc_ipindiscards, 242
 - snmp_inc_ipinhderrors, 242
 - snmp_inc_ipinreceives, 242
 - snmp_inc_ipinunknownprotos, 242
 - snmp_inc_ipoutdiscards, 242
 - snmp_inc_ipoutnoroutes, 243
 - snmp_inc_ipoutrequests, 243
 - snmp_inc_ipreasmlfails, 243
 - snmp_inc_ipreasmloks, 243
 - snmp_inc_ipreasmlreqds, 243
 - snmp_inc_iproutingdiscards, 243
 - snmp_inc_snmpinbadcommunitynames, 243
 - snmp_inc_snmpinbadcommunityuses, 243
 - snmp_inc_snmpinbadvalues, 243
 - snmp_inc_snmpinbadversions, 243
 - snmp_inc_snmpingenerrs, 243
 - snmp_inc_snmpingetnexts, 244
 - snmp_inc_snmpingetrequests, 244
 - snmp_inc_snmpingetresponses, 244
 - snmp_inc_snmpinnosuchnames, 244
 - snmp_inc_snmpinpkts, 244
 - snmp_inc_snmpinreadonllys, 244
 - snmp_inc_snmpinsetrequests, 244
 - snmp_inc_snmpintoobigs, 244

- snmp_inc_snmpintraps, 244
- snmp_inc_snmpoutbadvalues, 244
- snmp_inc_snmpoutgenerrs, 244
- snmp_inc_snmpoutgetnexts, 244
- snmp_inc_snmpoutgetrequests, 245
- snmp_inc_snmpoutgetresponses, 245
- snmp_inc_snmpoutnosuchnames, 245
- snmp_inc_snmpoutpkts, 245
- snmp_inc_snmpoutsetrequests, 245
- snmp_inc_snmpouttoobigs, 245
- snmp_inc_snmpouttraps, 245
- snmp_inc_sysuptime, 245
- snmp_inc_tcpactiveopens, 245
- snmp_inc_tcpattemptfails, 245
- snmp_inc_tcestabresets, 245
- snmp_inc_tcpinerrs, 245
- snmp_inc_tcpinsegs, 246
- snmp_inc_tcpoutrst, 246
- snmp_inc_tcpoutsegs, 246
- snmp_inc_tcppassiveopens, 246
- snmp_inc_tcpretranssegs, 246
- snmp_inc_udpindatagrams, 246
- snmp_inc_udpinerrors, 246
- snmp_inc_udpnoports, 246
- snmp_inc_udpoutdatagrams, 246
- snmp_insert_arpidx_tree, 246
- snmp_insert_ipaddridx_tree, 246
- snmp_insert_iprteidx_tree, 246
- snmp_insert_udpidx_tree, 247
- snmp_set_snmpenableauthentraps, 247
- snmp_set_syscontact, 247
- snmp_set_sysdescr, 247
- snmp_set_syslocation, 247
- snmp_set_sysname, 247
- snmp_set_sysobjid, 247
- snmp_add_ifinOctets
 - snmp.h, 238
- snmp_add_ifoutOctets
 - snmp.h, 238
- snmp_add_snmpintotalreqvars
 - snmp.h, 238
- snmp_add_snmpintotalsetvars
 - snmp.h, 238
- snmp_add_sysuptime
 - snmp.h, 238
- snmp_dec_iflist
 - snmp.h, 238
- snmp_delete_arpidx_tree
 - snmp.h, 238
- snmp_delete_ipaddridx_tree
 - snmp.h, 238
- snmp_delete_iprteidx_tree
 - snmp.h, 238
- snmp_delete_udpidx_tree
 - snmp.h, 238
- snmp_get_snmpenableauthentraps
 - snmp.h, 239
- snmp_get_snmpgrpid_ptr
 - snmp.h, 239
- snmp_get_sysobjid_ptr
 - snmp.h, 239
- snmp_get_sysuptime
 - snmp.h, 239
- snmp_ifType
 - snmp.h, 247
- snmp_ifType_basicISDN
 - snmp.h, 248
- snmp_ifType_ddn_x25
 - snmp.h, 247
- snmp_ifType_ds1
 - snmp.h, 248
- snmp_ifType_ds3
 - snmp.h, 248
- snmp_ifType_e1
 - snmp.h, 248
- snmp_ifType_eon
 - snmp.h, 248
- snmp_ifType_ethernet_3Mbit
 - snmp.h, 248
- snmp_ifType_ethernet_csmacd
 - snmp.h, 247
- snmp_ifType_fddi
 - snmp.h, 248
- snmp_ifType_frame_relay
 - snmp.h, 248
- snmp_ifType_hdlc1822
 - snmp.h, 247
- snmp_ifType_hyperchannel
 - snmp.h, 248
- snmp_ifType_iso88023_csmacd
 - snmp.h, 247
- snmp_ifType_iso88024_tokenBus
 - snmp.h, 247
- snmp_ifType_iso88025_tokenRing
 - snmp.h, 247
- snmp_ifType_iso88026_man
 - snmp.h, 248
- snmp_ifType_lapb
 - snmp.h, 248
- snmp_ifType_nsip
 - snmp.h, 248
- snmp_ifType_other
 - snmp.h, 247
- snmp_ifType_ppp
 - snmp.h, 248
- snmp_ifType_primaryISDN
 - snmp.h, 248
- snmp_ifType_propPointToPointSerial
 - snmp.h, 248
- snmp_ifType_proteon_10Mbit
 - snmp.h, 248
- snmp_ifType_proteon_80Mbit
 - snmp.h, 248
- snmp_ifType_regular1822
 - snmp.h, 247
- snmp_ifType_rfc877_x25

- snmp.h, [247](#)
- snmp_ifType_sdLc
 - snmp.h, [248](#)
- snmp_ifType_sIp
 - snmp.h, [248](#)
- snmp_ifType_slIp
 - snmp.h, [248](#)
- snmp_ifType_softwareLoopback
 - snmp.h, [248](#)
- snmp_ifType_starLan
 - snmp.h, [248](#)
- snmp_ifType_ultra
 - snmp.h, [248](#)
- snmp_inc_icmpinaddrmaskreps
 - snmp.h, [239](#)
- snmp_inc_icmpinaddrmasks
 - snmp.h, [239](#)
- snmp_inc_icmpindestunreachs
 - snmp.h, [239](#)
- snmp_inc_icmpinechoreps
 - snmp.h, [239](#)
- snmp_inc_icmpinechos
 - snmp.h, [239](#)
- snmp_inc_icmpinerrors
 - snmp.h, [239](#)
- snmp_inc_icmpinmsgs
 - snmp.h, [239](#)
- snmp_inc_icmpinparmprobs
 - snmp.h, [239](#)
- snmp_inc_icmpinredirects
 - snmp.h, [240](#)
- snmp_inc_icmpinsrcquenchs
 - snmp.h, [240](#)
- snmp_inc_icmpintimeexcds
 - snmp.h, [240](#)
- snmp_inc_icmpintimestamppreps
 - snmp.h, [240](#)
- snmp_inc_icmpintimestamps
 - snmp.h, [240](#)
- snmp_inc_icmpoutaddrmaskreps
 - snmp.h, [240](#)
- snmp_inc_icmpoutaddrmasks
 - snmp.h, [240](#)
- snmp_inc_icmpoutdestunreachs
 - snmp.h, [240](#)
- snmp_inc_icmpoutechoreps
 - snmp.h, [240](#)
- snmp_inc_icmpoutechos
 - snmp.h, [240](#)
- snmp_inc_icmpoutererrors
 - snmp.h, [240](#)
- snmp_inc_icmpoutmsgs
 - snmp.h, [240](#)
- snmp_inc_icmpoutparmprobs
 - snmp.h, [241](#)
- snmp_inc_icmpoutredirects
 - snmp.h, [241](#)
- snmp_inc_icmpoutsrcquenchs
 - snmp.h, [241](#)
- snmp_inc_icmpouttimeexcds
 - snmp.h, [241](#)
- snmp_inc_icmpouttimestamppreps
 - snmp.h, [241](#)
- snmp_inc_icmpouttimestamps
 - snmp.h, [241](#)
- snmp_inc_ifindiscards
 - snmp.h, [241](#)
- snmp_inc_ifinnucastpkts
 - snmp.h, [241](#)
- snmp_inc_ifinucastpkts
 - snmp.h, [241](#)
- snmp_inc_iflist
 - snmp.h, [241](#)
- snmp_inc_ifoutdiscards
 - snmp.h, [241](#)
- snmp_inc_ifoutnucastpkts
 - snmp.h, [241](#)
- snmp_inc_ifoutucastpkts
 - snmp.h, [242](#)
- snmp_inc_ipforwdatagrams
 - snmp.h, [242](#)
- snmp_inc_ipfragcreates
 - snmp.h, [242](#)
- snmp_inc_ipfragfails
 - snmp.h, [242](#)
- snmp_inc_ipfragoks
 - snmp.h, [242](#)
- snmp_inc_ipinaddrerrors
 - snmp.h, [242](#)
- snmp_inc_ipindelivers
 - snmp.h, [242](#)
- snmp_inc_ipindiscards
 - snmp.h, [242](#)
- snmp_inc_ipinhdrerrors
 - snmp.h, [242](#)
- snmp_inc_ipinreceives
 - snmp.h, [242](#)
- snmp_inc_ipinunknownprotos
 - snmp.h, [242](#)
- snmp_inc_ipoutdiscards
 - snmp.h, [242](#)
- snmp_inc_ipoutnoroutes
 - snmp.h, [243](#)
- snmp_inc_ipoutrequests
 - snmp.h, [243](#)
- snmp_inc_ipreamfails
 - snmp.h, [243](#)
- snmp_inc_ipreamoks
 - snmp.h, [243](#)
- snmp_inc_ipreamreqds
 - snmp.h, [243](#)
- snmp_inc_iproutingdiscards
 - snmp.h, [243](#)
- snmp_inc_snmpinasnparseerrs
 - snmp.h, [243](#)
- snmp_inc_snmpinbadcommunitynames

snmp.h, [243](#)
snmp_inc_snmpinbadcommunityuses
snmp.h, [243](#)
snmp_inc_snmpinbadvalues
snmp.h, [243](#)
snmp_inc_snmpinbadversions
snmp.h, [243](#)
snmp_inc_snmpingenerrs
snmp.h, [243](#)
snmp_inc_snmpingetnexts
snmp.h, [244](#)
snmp_inc_snmpingetrequests
snmp.h, [244](#)
snmp_inc_snmpingetresponses
snmp.h, [244](#)
snmp_inc_snmpinnosuchnames
snmp.h, [244](#)
snmp_inc_snmpinpkts
snmp.h, [244](#)
snmp_inc_snmpinreadonlys
snmp.h, [244](#)
snmp_inc_snmpinsetrequests
snmp.h, [244](#)
snmp_inc_snmpintoobigs
snmp.h, [244](#)
snmp_inc_snmpintraps
snmp.h, [244](#)
snmp_inc_snmpoutbadvalues
snmp.h, [244](#)
snmp_inc_snmpoutgenerrs
snmp.h, [244](#)
snmp_inc_snmpoutgetnexts
snmp.h, [244](#)
snmp_inc_snmpoutgetrequests
snmp.h, [245](#)
snmp_inc_snmpoutgetresponses
snmp.h, [245](#)
snmp_inc_snmpoutnosuchnames
snmp.h, [245](#)
snmp_inc_snmpoutpkts
snmp.h, [245](#)
snmp_inc_snmpoutsetrequests
snmp.h, [245](#)
snmp_inc_snmpouttoobigs
snmp.h, [245](#)
snmp_inc_snmpouttraps
snmp.h, [245](#)
snmp_inc_sysuptime
snmp.h, [245](#)
snmp_inc_tcpactiveopens
snmp.h, [245](#)
snmp_inc_tcpattemptfails
snmp.h, [245](#)
snmp_inc_tcpstabresets
snmp.h, [245](#)
snmp_inc_tcpinerrs
snmp.h, [245](#)
snmp_inc_tcpinsegs
snmp.h, [246](#)
snmp_inc_tcpoutrst
snmp.h, [246](#)
snmp_inc_tcpoutsegs
snmp.h, [246](#)
snmp_inc_tcppassiveopens
snmp.h, [246](#)
snmp_inc_tcpretranssegs
snmp.h, [246](#)
snmp_inc_udpindatagrams
snmp.h, [246](#)
snmp_inc_udpinerrors
snmp.h, [246](#)
snmp_inc_udpnoports
snmp.h, [246](#)
snmp_inc_udpoutdatagrams
snmp.h, [246](#)
snmp_insert_arpidx_tree
snmp.h, [246](#)
snmp_insert_ipaddridx_tree
snmp.h, [246](#)
snmp_insert_iprtidx_tree
snmp.h, [246](#)
snmp_insert_udpidx_tree
snmp.h, [247](#)
snmp_set_snmpenableauthtraps
snmp.h, [247](#)
snmp_set_syscontact
snmp.h, [247](#)
snmp_set_sysdescr
snmp.h, [247](#)
snmp_set_syslocation
snmp.h, [247](#)
snmp_set_sysname
snmp.h, [247](#)
snmp_set_sysobjid
snmp.h, [247](#)
src/api/api_lib.c, [41](#)
src/api/api_msg.c, [42](#)
src/api/err.c, [43](#)
src/api/hetbuf.c, [44](#)
src/api/hetdb.c, [45](#)
src/api/hetifapi.c, [46](#)
src/api/sockets.c, [47](#)
src/api/tcpip.c, [47](#)
src/core/def.c, [50](#)
src/core/dhcp.c, [53](#)
src/core/dns.c, [54](#)
src/core/init.c, [55](#)
src/core/ipv4/autoip.c, [56](#)
src/core/ipv4/icmp.c, [57](#)
src/core/ipv4/igmp.c, [58](#)
src/core/ipv4/inet.c, [58](#)
src/core/ipv4/inet_chksum.c, [59](#)
src/core/ipv4/ip.c, [62](#)
src/core/ipv4/ip_addr.c, [65](#)
src/core/ipv4/ip_frag.c, [70](#)
src/core/ipv6/icmp6.c, [71](#)

- src/core/ipv6/inet6.c, 71
- src/core/ipv6/ip6.c, 73
- src/core/ipv6/ip6_addr.c, 74
- src/core/mem.c, 76
- src/core/memp.c, 79
- src/core/netif.c, 82
- src/core/pbuf.c, 87
- src/core/raw.c, 94
- src/core/snmp/asn1_dec.c, 96
- src/core/snmp/asn1_enc.c, 96
- src/core/snmp/mib2.c, 97
- src/core/snmp/mib_structs.c, 98
- src/core/snmp/msg_in.c, 100
- src/core/snmp/msg_out.c, 101
- src/core/stats.c, 101
- src/core/sys.c, 102
- src/core/tcp.c, 104
- src/core/tcp_in.c, 104
- src/core/tcp_out.c, 105
- src/core/timers.c, 106
- src/core/udp.c, 108
- src/include/ipv4/lwip/autoip.h, 109
- src/include/ipv4/lwip/icmp.h, 110
- src/include/ipv4/lwip/igmp.h, 114
- src/include/ipv4/lwip/inet.h, 116
- src/include/ipv4/lwip/inet_chksum.h, 122
- src/include/ipv4/lwip/ip.h, 125
- src/include/ipv4/lwip/ip_addr.h, 138
- src/include/ipv4/lwip/ip_frag.h, 151
- src/include/ipv6/lwip/icmp.h, 113
- src/include/ipv6/lwip/inet.h, 120
- src/include/ipv6/lwip/ip.h, 134
- src/include/ipv6/lwip/ip_addr.h, 149
- src/include/lwip/api.h, 153
- src/include/lwip/api_msg.h, 154
- src/include/lwip/arch.h, 155
- src/include/lwip/debug.h, 156
- src/include/lwip/def.h, 159
- src/include/lwip/dhcp.h, 162
- src/include/lwip/dns.h, 162
- src/include/lwip/err.h, 163
- src/include/lwip/init.h, 166
- src/include/lwip/mem.h, 169
- src/include/lwip/memp.h, 173
- src/include/lwip/memp_std.h, 175
- src/include/lwip/netbuf.h, 176
- src/include/lwip/netdb.h, 179
- src/include/lwip/netif.h, 180
- src/include/lwip/netifapi.h, 189
- src/include/lwip/opt.h, 190
- src/include/lwip/pbuf.h, 221
- src/include/lwip/raw.h, 230
- src/include/lwip/sio.h, 230
- src/include/lwip/snmp.h, 234
- src/include/lwip/snmp_asn1.h, 248
- src/include/lwip/snmp_msg.h, 249
- src/include/lwip/snmp_structs.h, 251
- src/include/lwip/sockets.h, 252
- src/include/lwip/stats.h, 253
- src/include/lwip/sys.h, 258
- src/include/lwip/tcp.h, 266
- src/include/lwip/tcp_impl.h, 267
- src/include/lwip/tcpip.h, 267
- src/include/lwip/timers.h, 272
- src/include/lwip/udp.h, 275
- src/include/netif/etharp.h, 275
- src/include/netif/ppp_oe.h, 276
- src/include/netif/slipif.h, 277
- src/include/posix/netdb.h, 179
- src/include/posix/sys/socket.h, 279
- src/netif/etharp.c, 281
- src/netif/ethernetif.c, 281
- src/netif/ppp/auth.c, 283
- src/netif/ppp/auth.h, 283
- src/netif/ppp/chap.c, 284
- src/netif/ppp/chap.h, 285
- src/netif/ppp/chpms.c, 289
- src/netif/ppp/chpms.h, 289
- src/netif/ppp/fsm.c, 290
- src/netif/ppp/fsm.h, 290
- src/netif/ppp/ipcp.c, 294
- src/netif/ppp/ipcp.h, 294
- src/netif/ppp/lcp.c, 297
- src/netif/ppp/lcp.h, 297
- src/netif/ppp/magic.c, 301
- src/netif/ppp/magic.h, 301
- src/netif/ppp/md5.c, 301
- src/netif/ppp/md5.h, 302
- src/netif/ppp/pap.c, 302
- src/netif/ppp/pap.h, 303
- src/netif/ppp/ppp.c, 303
- src/netif/ppp/ppp.h, 304
- src/netif/ppp/ppp_impl.h, 305
- src/netif/ppp/ppp_oe.c, 306
- src/netif/ppp/pppdebug.h, 307
- src/netif/ppp/randm.c, 309
- src/netif/ppp/randm.h, 309
- src/netif/ppp/vj.c, 310
- src/netif/ppp/vj.h, 310
- src/netif/slipif.c, 314
- starting
 - fsm_callbacks, 16
- state
 - fsm, 14
 - netif, 32
- stats.h
 - ETHARP_STATS_DISPLAY, 254
 - ETHARP_STATS_INC, 254
 - ICMP_STATS_DISPLAY, 254
 - ICMP_STATS_INC, 254
 - IGMP_STATS_DISPLAY, 254
 - IGMP_STATS_INC, 255
 - IP_STATS_DISPLAY, 255
 - IP_STATS_INC, 255
 - IPFRAG_STATS_DISPLAY, 255
 - IPFRAG_STATS_INC, 255

- LINK_STATS_DISPLAY, 255
- LINK_STATS_INC, 255
- MEM_STATS_AVAIL, 255
- MEM_STATS_DEC_USED, 255
- MEM_STATS_DISPLAY, 255
- MEM_STATS_INC, 255
- MEM_STATS_INC_USED, 255
- MEMP_STATS_AVAIL, 256
- MEMP_STATS_DEC, 256
- MEMP_STATS_DISPLAY, 256
- MEMP_STATS_INC, 256
- MEMP_STATS_INC_USED, 256
- STATS_DEC, 256
- STATS_INC, 257
- STATS_INC_USED, 257
- SYS_STATS_DEC, 257
- SYS_STATS_DISPLAY, 257
- SYS_STATS_INC, 257
- SYS_STATS_INC_USED, 257
- stats_display, 256
- stats_display_igmp, 256
- stats_display_mem, 256
- stats_display_memp, 256
- stats_display_proto, 256
- stats_display_sys, 256
- stats_init, 257
- TCP_STATS_DISPLAY, 257
- TCP_STATS_INC, 257
- UDP_STATS_DISPLAY, 257
- UDP_STATS_INC, 257
- stats_display
 - stats.h, 256
- stats_display_igmp
 - stats.h, 256
- stats_display_mem
 - stats.h, 256
- stats_display_memp
 - stats.h, 256
- stats_display_proto
 - stats.h, 256
- stats_display_sys
 - stats.h, 256
- stats_init
 - stats.h, 257
- sys.c
 - sys_msleep, 103
- sys.h
 - lwip_thread_fn, 261
 - SYS_ARCH_DEC, 259
 - SYS_ARCH_DECL_PROTECT, 260
 - SYS_ARCH_GET, 260
 - SYS_ARCH_INC, 260
 - SYS_ARCH_PROTECT, 260
 - SYS_ARCH_SET, 260
 - SYS_ARCH_TIMEOUT, 260
 - SYS_ARCH_UNPROTECT, 261
 - SYS_MBOX_EMPTY, 261
 - sys_arch_mbox_fetch, 261
 - sys_arch_mbox_tryfetch, 262
 - sys_arch_sem_wait, 262
 - sys_init, 262
 - sys_jiffies, 262
 - sys_mbox_fetch, 261
 - sys_mbox_free, 262
 - sys_mbox_new, 262
 - sys_mbox_post, 263
 - sys_mbox_set_invalid, 263
 - sys_mbox_tryfetch, 261
 - sys_mbox_trypost, 263
 - sys_mbox_valid, 263
 - sys_msleep, 263
 - sys_mutex_free, 263
 - sys_mutex_lock, 264
 - sys_mutex_new, 264
 - sys_mutex_set_invalid, 264
 - sys_mutex_unlock, 264
 - sys_mutex_valid, 264
 - sys_now, 264
 - sys_sem_free, 264
 - sys_sem_new, 265
 - sys_sem_set_invalid, 265
 - sys_sem_signal, 265
 - sys_sem_valid, 265
 - sys_sem_wait, 261
 - sys_thread_new, 265
- sys_arch_mbox_fetch
 - sys.h, 261
- sys_arch_mbox_tryfetch
 - sys.h, 262
- sys_arch_sem_wait
 - sys.h, 262
- sys_init
 - sys.h, 262
- sys_jiffies
 - sys.h, 262
- sys_mbox_fetch
 - sys.h, 261
- sys_mbox_free
 - sys.h, 262
- sys_mbox_new
 - sys.h, 262
- sys_mbox_post
 - sys.h, 263
- sys_mbox_set_invalid
 - sys.h, 263
- sys_mbox_tryfetch
 - sys.h, 261
- sys_mbox_trypost
 - sys.h, 263
- sys_mbox_valid
 - sys.h, 263
- sys_msleep
 - sys.c, 103
 - sys.h, 263
- sys_mutex_free
 - sys.h, 263

- sys_mutex_lock
 - sys.h, [264](#)
- sys_mutex_new
 - sys.h, [264](#)
- sys_mutex_set_invalid
 - sys.h, [264](#)
- sys_mutex_unlock
 - sys.h, [264](#)
- sys_mutex_valid
 - sys.h, [264](#)
- sys_now
 - sys.h, [264](#)
- sys_sem_free
 - sys.h, [264](#)
- sys_sem_new
 - sys.h, [265](#)
- sys_sem_set_invalid
 - sys.h, [265](#)
- sys_sem_signal
 - sys.h, [265](#)
- sys_sem_valid
 - sys.h, [265](#)
- sys_sem_wait
 - sys.h, [261](#)
- sys_thread_new
 - sys.h, [265](#)
- sys_timeo, [33](#)
 - arg, [34](#)
 - h, [34](#)
 - next, [34](#)
 - time, [34](#)
- sys_timeout
 - timers.h, [274](#)
- sys_timeout_handler
 - timers.h, [274](#)
- sys_timeouts_init
 - timers.h, [274](#)
- sys_timeouts_mbox_fetch
 - timers.h, [274](#)
- sys_untimeout
 - timers.h, [274](#)
- TCP_CALCULATE_EFF_SEND_MSS
 - opt.h, [216](#)
- TCP_CWND_DEBUG
 - opt.h, [217](#)
- TCP_DEBUG
 - opt.h, [217](#)
- TCP_DEFAULT_LISTEN_BACKLOG
 - opt.h, [217](#)
- TCP_FR_DEBUG
 - opt.h, [217](#)
- TCP_INPUT_DEBUG
 - opt.h, [217](#)
- TCP_LISTEN_BACKLOG
 - opt.h, [217](#)
- TCP_MAXRTX
 - opt.h, [217](#)
- TCP_MSS
 - opt.h, [217](#)
- TCP_OOSEQ_MAX_BYTES
 - opt.h, [218](#)
- TCP_OOSEQ_MAX_PBUFS
 - opt.h, [218](#)
- TCP_OUTPUT_DEBUG
 - opt.h, [218](#)
- TCP_OVERSIZE
 - opt.h, [218](#)
- TCP_PUSH_BIT
 - vj.h, [313](#)
- TCP_QLEN_DEBUG
 - opt.h, [218](#)
- TCP_QUEUE_OOSEQ
 - opt.h, [218](#)
- TCP_RST_DEBUG
 - opt.h, [218](#)
- TCP_RTO_DEBUG
 - opt.h, [218](#)
- TCP_SND_BUF
 - opt.h, [219](#)
- TCP_SND_QUEUELEN
 - opt.h, [219](#)
- TCP_SNDLOWAT
 - opt.h, [219](#)
- TCP_SNDQUEUELOWAT
 - opt.h, [219](#)
- TCP_STATS
 - opt.h, [219](#)
- TCP_STATS_DISPLAY
 - stats.h, [257](#)
- TCP_STATS_INC
 - stats.h, [257](#)
- TCP_SYNMAXRTX
 - opt.h, [219](#)
- TCP_TTL
 - opt.h, [219](#)
- TCP_WND
 - opt.h, [219](#)
- TCP_WND_DEBUG
 - opt.h, [219](#)
- TCP_WND_UPDATE_THRESHOLD
 - opt.h, [220](#)
- TCPIP_APIMSG
 - tcip.h, [269](#)
- TCPIP_APIMSG_ACK
 - tcip.h, [269](#)
- TCPIP_DEBUG
 - opt.h, [220](#)
- TCPIP_MBOX_SIZE
 - opt.h, [220](#)
- TCPIP_MSG_CALLBACK
 - tcip.h, [270](#)
- TCPIP_MSG_CALLBACK_STATIC
 - tcip.h, [270](#)
- TCPIP_MSG_INPKT
 - tcip.h, [270](#)
- TCPIP_NETIFAPI

- tcpip.h, [269](#)
- TCPIP_NETIFAPI_ACK
 - tcpip.h, [269](#)
- TCPIP_THREAD_NAME
 - opt.h, [220](#)
- TCPIP_THREAD_PRIO
 - opt.h, [220](#)
- TCPIP_THREAD_STACKSIZE
 - opt.h, [220](#)
- TERMACK
 - fsm.h, [293](#)
- TERMREQ
 - fsm.h, [293](#)
- TIMERS_DEBUG
 - opt.h, [220](#)
- TRACELCP
 - pppdebug.h, [308](#)
- TYPE_COMPRESSED_TCP
 - vj.h, [313](#)
- TYPE_ERROR
 - vj.h, [313](#)
- TYPE_IP
 - vj.h, [313](#)
- TYPE_UNCOMPRESSED_TCP
 - vj.h, [313](#)
- tclass1
 - ip_hdr, [21](#)
- tclass2
 - ip_hdr, [21](#)
- tcp_timer_needed
 - timers.c, [107](#)
- tcpip.c
 - mem_free_callback, [48](#)
 - pbuf_free_callback, [48](#)
 - tcpip_callback_with_block, [49](#)
 - tcpip_callbackmsg_delete, [49](#)
 - tcpip_callbackmsg_new, [49](#)
 - tcpip_init, [49](#)
 - tcpip_input, [50](#)
 - tcpip_trycallback, [50](#)
- tcpip.h
 - LOCK_TCPIP_CORE, [269](#)
 - LWIP_TCPIP_THREAD_ALIVE, [269](#)
 - mem_free_callback, [270](#)
 - pbuf_free_callback, [270](#)
 - TCPIP_APIMSG, [269](#)
 - TCPIP_APIMSG_ACK, [269](#)
 - TCPIP_MSG_CALLBACK, [270](#)
 - TCPIP_MSG_CALLBACK_STATIC, [270](#)
 - TCPIP_MSG_INPKT, [270](#)
 - TCPIP_NETIFAPI, [269](#)
 - TCPIP_NETIFAPI_ACK, [269](#)
 - tcpip_callback, [269](#)
 - tcpip_callback_fn, [270](#)
 - tcpip_callback_with_block, [271](#)
 - tcpip_callbackmsg_delete, [271](#)
 - tcpip_callbackmsg_new, [271](#)
 - tcpip_init, [271](#)
 - tcpip_init_done_fn, [270](#)
 - tcpip_input, [272](#)
 - tcpip_init_done_fn, [270](#)
 - tcpip_input, [272](#)
 - tcpip_msg, [34](#)
 - cb, [35](#)
 - ctx, [35](#)
 - function, [35](#)
 - inp, [36](#)
 - msg, [36](#)
 - netif, [36](#)
 - p, [36](#)
 - sem, [36](#)
 - type, [36](#)
 - tcpip_msg_type, [270](#)
 - tcpip_trycallback, [272](#)
 - term_reason, [14](#)
 - term_reason_len, [14](#)
 - time, [34](#)
 - timeouttime, [11](#)
 - timers.c, [107](#)
 - timers.h, [274](#)

- sys_timeouts_mbox_fetch, 274
 - sys_untimeout, 274
- tot_len
 - pbuf, 33
- tstate
 - vjcompress, 38
- type
 - pbuf, 33
 - tcpip_msg, 36
- UDP_DEBUG
 - opt.h, 220
- UDP_STATS
 - opt.h, 220
- UDP_STATS_DISPLAY
 - stats.h, 257
- UDP_STATS_INC
 - stats.h, 257
- UDP_TTL
 - opt.h, 221
- UNLOCK_TCPIP_CORE
 - tcpip.h, 269
- UPAPDEBUG
 - pppdebug.h, 309
- USE_CRYPT
 - chpms.c, 289
- unit
 - chap_state, 11
 - fsm, 15
- up
 - fsm_callbacks, 16
- used
 - mem, 28
- v
 - ip_hdr, 21
- VJF_TOSS
 - vj.h, 313
- vj.h
 - cs_hdr, 312
 - cs_ip, 312
 - MAX_HDR, 312
 - MAX_SLOTS, 312
 - NEW_A, 312
 - NEW_C, 312
 - NEW_I, 312
 - NEW_S, 312
 - NEW_U, 312
 - NEW_W, 313
 - SPECIAL_D, 313
 - SPECIAL_I, 313
 - SPECIALS_MASK, 313
 - TCP_PUSH_BIT, 313
 - TYPE_COMPRESSED_TCP, 313
 - TYPE_ERROR, 313
 - TYPE_IP, 313
 - TYPE_UNCOMPRESSED_TCP, 313
 - VJF_TOSS, 313
 - vj_compress_init, 313
 - vj_compress_tcp, 313
 - vj_uncompress_err, 313
 - vj_uncompress_tcp, 314
 - vj_uncompress_uncomp, 314
- vj_compress_init
 - vj.h, 313
- vj_compress_tcp
 - vj.h, 313
- vj_protocol
 - ipcp_options, 24
- vj_uncompress_err
 - vj.h, 313
- vj_uncompress_tcp
 - vj.h, 314
- vj_uncompress_uncomp
 - vj.h, 314
- vjcompress, 36
 - compressSlot, 37
 - flags, 37
 - last_cs, 37
 - last_recv, 38
 - last_xmit, 38
 - maxSlotIndex, 38
 - rstate, 38
 - tstate, 38
- vjcs_u
 - cstate, 12
- vjs_compressed
 - vjstat, 39
- vjs_compressedin
 - vjstat, 39
- vjs_errorin
 - vjstat, 39
- vjs_misses
 - vjstat, 39
- vjs_packets
 - vjstat, 39
- vjs_searches
 - vjstat, 39
- vjs_tossed
 - vjstat, 39
- vjs_uncompressedin
 - vjstat, 39
- vjstat, 38
 - vjs_compressed, 39
 - vjs_compressedin, 39
 - vjs_errorin, 39
 - vjs_misses, 39
 - vjs_packets, 39
 - vjs_searches, 39
 - vjs_tossed, 39
 - vjs_uncompressedin, 39
- winsaddr
 - ipcp_options, 24
- X8_F
 - arch.h, 156
- xmit_accm

lcp.h, [301](#)