
IpWay/RT free Tcp/Ip Stack Library - User Manual

Contents

1 Revision History	6
2 Introduction	7
2.1 Overview	7
2.2 Acronyms and Definitions	7
2.3 Reference List	8
3 Library	10
3.1 Library Design Summary	10
3.2 Limitations	10
3.3 Legend	10
3.4 Software specification	11
3.4.1 Configuration parameters	11
3.4.2 Macros Reference	18
3.4.3 Enum Reference	22
3.4.4 Function Reference	24
3.4.5 Structs Reference	42
3.4.6 Types Reference	46
3.4.7 Variables Reference	46
3.5 Binding	47
3.5.1 Ethernet Interface	47
3.5.2 Socket Interface	47
3.5.3 Timers and Scheduler	48
4 Applications	49
4.1 Web Server	49
5 Disclaimer	50

List of tables

Table 1. Revision History	6
Table 2. Acronyms and Definitions	11
Table 3. Reference List	8
Table 4. Library limitations:	10
Table 5. Legend of code entity	11
Table 6. Define IWD_CFG_ARP_ENABLED Description	11
Table 7. Define IWD_CFG_ARP_NUM_GRATUITOUS_ARP_REPLIES Description	11
Table 8. Define IWD_CFG_ARP_TABLE_ENTRY_TIMEOUT Description	11
Table 9. Define IWD_CFG_ARP_TABLE_SIZE Description	12
Table 10. Define IWD_CFG_AUTO_IP_TIMEOUT Description	12
Table 11. Define IWD_CFG_AUTOIP_ENABLED Description	12
Table 12. Define IWD_CFG_BUFFER_MEMORY_SIZE Description	12
Table 13. Define IWD_CFG_CPU_ARCHITECTURE Description	12
Table 14. Define IWD_CFG_DEBUG_ERROR Description	13
Table 15. Define IWD_CFG_DHCP_CLIENT_ENABLED Description	13
Table 16. Define IWD_CFG_DHCP_HOSTNAME_LEN Description	13
Table 17. Define IWD_CFG_DHCP_HOSTNAME_OPT_ENABLED Description	13
Table 18. Define IWD_CFG_DHCP_HOSTNAME_VALUE Description	13
Table 19. Define IWD_CFG_DHCP_LEASE_OPT_ENABLED Description	14
Table 20. Define IWD_CFG_DHCP_MAX_MSG_SIZE_OPT_ENABLED Description	14
Table 21. Define IWD_CFG_DHCP_MAX_MSG_SIZE_VALUE Description	14
Table 22. Define IWD_CFG_DHCP_PARAMETER_REQ_LIST_LEN Description	14
Table 23. Define IWD_CFG_ICMP_ENABLED Description	15
Table 24. Define IWD_CFG_ICMP_TTL Description	15
Table 25. Define IWD_CFG_INET6 Description	15
Table 26. Define IWD_CFG_MAX_NETID_SUPPORTED Description	15
Table 27. Define IWD_CFG_PATHMTU_DISCOVERY_SUPPORT Description	15
Table 28. Define IWD_CFG_SCHEDULER_PERIOD_MS Description	16
Table 29. Define IWD_CFG_SOCKET_MAX Description	16
Table 30. Define IWD_CFG_STATIC_IP_ADDR_GATEWAY_0 Description	16
Table 31. Define IWD_CFG_STATIC_SUBNET_MASK_NETIF_0 Description	16
Table 32. Define IWD_CFG_TCP_ENABLED Description	17
Table 33. Define IWD_CFG_TCP_SOCKET_MAX Description	17
Table 34. Define IWD_CFG_UDP_ENABLED Description	17
Table 35. Define IWD_CFG_UDP_SOCKET_MAX Description	17
Table 36. Define IWD_IP_ADDR_ASSIGN_MTD_AUTOIP_PRIO Description	17
Table 37. Define IWD_IP_ADDR_ASSIGN_MTD_STATIC_PRIO Description	18
Table 38. Define IWM_TIMER_GET_TIME Description	18
Table 39. IWM_MTX_NETID_LOCK Arguments	20
Table 40. IWM_MTX_NETID_UNLOCK Arguments	20
Table 41. IWM_MTX_SCKID_LOCK Arguments	20
Table 42. IWM_MTX_SCKID_UNLOCK Arguments	21
Table 43. IWM_MTX_SID_LOCK Arguments	21
Table 44. IWM_MTX_SID_UNLOCK Arguments	22
Table 45. Enumeration IWD_IpAddrAssignMtdType Values	22
Table 46. Enumeration IWD_IpAddrType Values	23
Table 47. Enumeration IWD_NetStateType Values	23

Table 48. Enumeration IWD_ProtocolType Values	23
Table 49. Enumeration IWD_TcpCopyType Values	24
Table 50. EIF_Cbk Arguments	24
Table 51. EIF_ChangeState Arguments	25
Table 52. EIF_ChangeState Return Values	25
Table 53. EIF_CtrlInit Arguments	25
Table 54. EIF_CtrlInit Return Values	26
Table 55. EIF_GetLinkStatus Arguments	26
Table 56. EIF_GetMACAddr Arguments	26
Table 57. EIF_GetState Arguments	27
Table 58. EIF_GetTxBuffer Arguments	27
Table 59. EIF_GetTxBuffer Return Values	28
Table 60. EIF_Init Arguments	28
Table 61. EIF_TxBuffer Arguments	29
Table 62. EIF_TxBuffer Return Values	29
Table 63. IWF_BindSocket Arguments	29
Table 64. IWF_BindSocket Return Values	30
Table 65. IWF_ChangeState Arguments	30
Table 66. IWF_ChangeState Return Values	31
Table 67. IWF_CloseSocket Arguments	31
Table 68. IWF_CloseSocket Return Values	31
Table 69. IWF_CreateSocket Arguments	32
Table 70. IWF_CreateSocket Return Values	32
Table 71. IWF_DhcpAddOption Arguments	32
Table 72. IWF_DhcpAddOption Return Values	33
Table 73. IWF_GetIpConfig Arguments	33
Table 74. IWF_GetIpConfig Return Values	33
Table 75. IWF_GetMACAddr Arguments	34
Table 76. IWF_GetMACAddr Return Values	34
Table 77. IWF_GetState Arguments	35
Table 78. IWF_IcmpSend Arguments	35
Table 79. IWF_IcmpSend Return Values	36
Table 80. IWF_Init Arguments	36
Table 81. IWF_TcpClose Arguments	37
Table 82. IWF_TcpClose Return Values	37
Table 83. IWF_TcpFreeRxBuffer Arguments	37
Table 84. IWF_TcpFreeRxBuffer Return Values	37
Table 85. IWF_TcpOpen Arguments	38
Table 86. IWF_TcpOpen Return Values	38
Table 87. IWF_TcpSend Arguments	39
Table 88. IWF_TcpSend Return Values	39
Table 89. IWF_UdpSend Arguments	40
Table 90. IWF_UdpSend Return Values	40
Table 91. SIF_Cbk Arguments	41
Table 92. SIF_TcpConnAccepted Arguments	42
Table 93. Structure IWD_IpAddrAssignCfgType member description	46
Table 94. Structure IWD_NetCfgType member description	46
Table 95. Structure IWD_NetIfType member description	46
Table 96. Structure IWD_SocketType member description	46

List of figures

Figure 1. IpWay Overview	7
Figure 2. Function EIF_Cbk References.	24
Figure 3. Inverse references of fuction EIF_GetTxBuffer	27
Figure 4. Inverse references of fuction EIF_TxBuffer	28
Figure 5. Inverse references of fuction IWF_DhcpAddOption	32
Figure 6. Inverse references of fuction IWF_TcpFreeRxBuffer	37
Figure 7. Inverse references of fuction IWF_UdpSend	40
Figure 8. Function SIF_Cbk References.	41
Figure 9. Struct IWD_IpAddrAssignCfgType	42
Figure 10. Struct IWD_NetCfgType	43
Figure 11. Struct IWD_SocketType	45

1 Revision History

Table 1: Revision History

Revision	Date	Author	Description
0.7.0	01-June-2015	G. Di Martino	Initial version
0.7.1	14-June-2015	G. Di Martino	Added Mutex
0.7.2	23-June-2015	G. Di Martino	Added new service IWF_GetState and new folder structure
0.7.3	12-July-2015	G. Di Martino	Added new service IWF_TimerUpdate and added a new parameter in the prototype of IWF_TcpSend function.
0.7.4	25-August-2015	G. Di Martino	Added new service EIF_GetLinkStatus; completed AU-TOIP service.

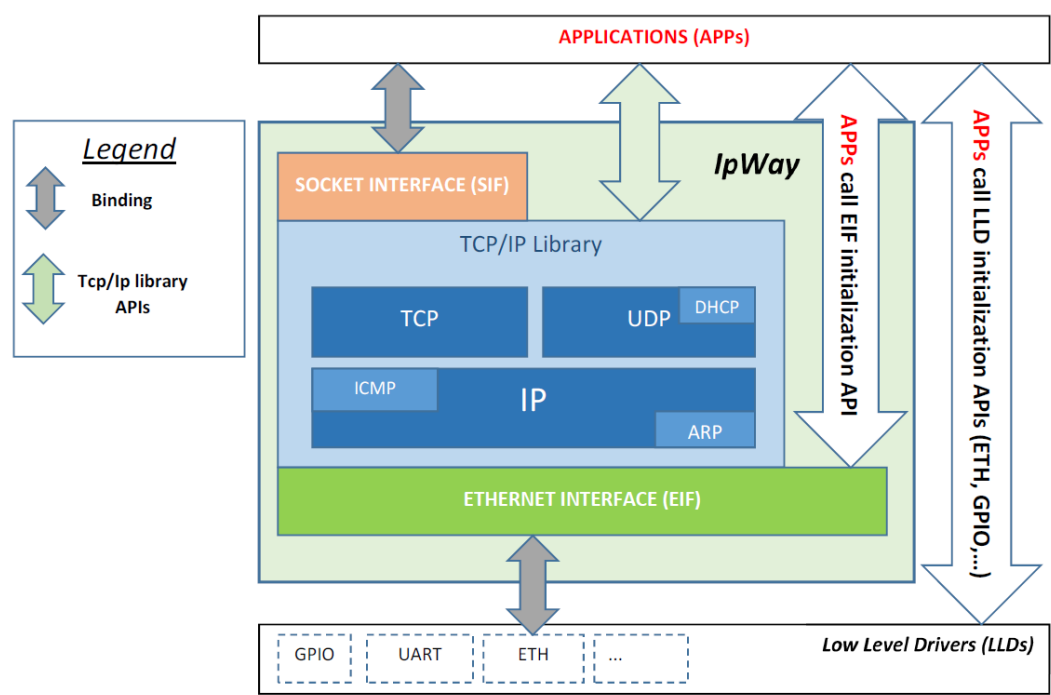
2 Introduction

This User Manual describes the IpWay/RT free Tcp/Ip Stack Library.

2.1 Overview

IpWay/RT is a tcp/ip stack library designed for real time embedded applications where execution efficiency and compact code are important requirements. IpWay is characterized by high portability, compact size and copyless feature also for application based on TCP protocol.

Figure 1: IpWay Overview



2.2 Acronyms and Definitions

Table 2: Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
C/CPP	C and C++ Source Code
FIFO	First In First Out
IETF	Internet Engineering Task Force

Term	Definition
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LSB	Least Significant Bit
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
TCP/IP	Transport Control Protocol/Internet Protocol
UDP	User Datagram Protocol
XML	Extensible Markup Language

2.3 Reference List

Table 3: Reference List

#	Title	Version
1	RFC 768 - User Datagram Protocol (UDP)	28 August 1980
2	RFC 790 - Assigned Numbers	September 1981
3	RFC 791 - Internet Protocol (IP)	September 1981
4	RFC 792 - Internet Control Message Protocol (ICMP)	September 1981
5	RFC 793 - Transmission Control Protocol (TCP)	September 1981
6	RFC 826 - Address Resolution Protocol (ARP)	November 1982
7	RFC 894 - Encapsulating Internet Protocol	April 1984
8	RFC 896 - Congestion Control in IP/TCP Internetworks (Nagle Algorithm)	6 January 1984
9	RFC 1122 - Requirements for Internet Hosts	October 1989
10	RFC 1191 - Path MTU Discovery	November 1990
11	RFC 2131 - Dynamic Host Configuration Protocol (DHCP)	March 1997

#	Title	Version
12	RFC 2132 - DHCP Options and BOOTP Vendor Extensions	March 1997

3 Library

3.1 Library Design Summary

IpWay library implements specific functionalities defined in below:

Address Resolution Protocol (ARP): Address Resolution Protocol (ARP) as defined in IETF RFC 826.

Internet Protocol (IP): Internet Protocol as defined in IETF RFC 791.

Dynamic Host Configuration Protocol (DHCP): Dynamic Host Configuration Protocol (DHCP) Client for the dynamic configuration of IPv4 addresses as defined in IETF RFC 2131 (Dynamic Host Configuration Protocol).

Internet Control Message Protocol (ICMP): The TcpIp supports the transmission and reception of Internet Control Message Protocol (ICMP) messages as defined in IETF RFC 792 (Internet Control Message Protocol in version 4).

User Datagram Protocol (UDP): The TcpIp implements the User Datagram Protocol (UDP) as defined in IETF RFC 768 (User Datagram Protocol).

Transmission Control Protocol (TCP): The TcpIp implements the Transmission Control Protocol (TCP) as defined in IETF RFC 793 (Transmission Control Protocol).

3.2 Limitations

This release of IpWay/RT free Tcp/Ip Stack Library has the following limitations:

Table 4: Library limitations:

Protocol	Limitation
IPv6	Not yet supported; this release supports IPv4 only. IPv6 protocol development is in working progress.
DHCP	Does not support DNS option in DHCP message
TCP	Window Scale Option is not sent in SYN message but is correctly read from SYN message of remote host and used by the library.
TCP	According to RFC 793, when a connection enters in the CLOSE WAIT state, the tcp/ip library should wait a timeout before sending the FIN message; this feature is not supported in this release, therefore the library send a FIN as soon as gets this state

3.3 Legend

All the entity in the source code (functions, defines, global variables and macros) start with **IW** which stands for **IpWay**, with **SI** which stands for **Socket Interface** and with **EI** which stands for **Ethernet Interface**. In adding to this prefix there is another one that indicates the nature of the entity; the following table reports the several cases:

Table 5: Legend of code entity

Prefix	Meaning
D_	D indicates a Define
F_	F indicates a Function
G_	G indicates a Global variable
M_	M indicates a Macro

Therefore an entity that starts with IWF_ is a function of IpWay Library, and so on.

3.4 Software specification

The following section contains the software specification.

3.4.1 Configuration parameters

List of parameters configurable for IpWay library.

3.4.1.1 Define IWD_CFG_ARP_ENABLED

ARP (Address Resolution Protocol).

Table 6: Define IWD_CFG_ARP_ENABLED Description

Name	IWD_CFG_ARP_ENABLED
Initializer	TRUE

3.4.1.2 Define IWD_CFG_ARP_NUM_GRATUITOUS_ARP_REPLIES

number of gratuitous ARP replies which shall be sent on assignment of a new IP address

Table 7: Define IWD_CFG_ARP_NUM_GRATUITOUS_ARP_REPLIES Description

Name	IWD_CFG_ARP_NUM_GRATUITOUS_ARP_REPLIES
Initializer	(iwui8)3

3.4.1.3 Define IWD_CFG_ARP_TABLE_ENTRY_TIMEOUT

Timeout [ms] after which an unused ARP entry is removed.

Table 8: Define IWD_CFG_ARP_TABLE_ENTRY_TIMEOUT Description

Name	IWD_CFG_ARP_TABLE_ENTRY_TIMEOUT
Initializer	(iwui32)600000

3.4.1.4 Define IWD_CFG_ARP_TABLE_SIZE

Maximum number of entries in the ARP table.

Table 9: Define IWD_CFG_ARP_TABLE_SIZE Description

Name	IWD_CFG_ARP_TABLE_SIZE
Initializer	(iwui16)0x000a

3.4.1.5 Define IWD_CFG_AUTO_IP_TIMEOUT

Timeout [ms] waiting for DHCP service; after which starts Auto-Ip methods.

Table 10: Define IWD_CFG_AUTO_IP_TIMEOUT Description

Name	IWD_CFG_AUTO_IP_TIMEOUT
Initializer	(iwui32)60000

3.4.1.6 Define IWD_CFG_AUTOIP_ENABLED

Auto-IP (automatic private IP addressing) sub-module.

Table 11: Define IWD_CFG_AUTOIP_ENABLED Description

Name	IWD_CFG_AUTOIP_ENABLED
Initializer	TRUE

3.4.1.7 Define IWD_CFG_BUFFER_MEMORY_SIZE

Memory size in bytes reserved for TCP/IP buffers.

Table 12: Define IWD_CFG_BUFFER_MEMORY_SIZE Description

Name	IWD_CFG_BUFFER_MEMORY_SIZE
Initializer	0x00015000UL

3.4.1.8 Define IWD_CFG_CPU_ARCHITECTURE

Cpu Architecture Type (byte order).

Table 13: Define IWD_CFG_CPU_ARCHITECTURE Description

Name	IWD_CFG_CPU_ARCHITECTURE
Initializer	IWD_BIG_ENDIAN

3.4.1.9 Define IWD_CFG_DEBUG_ERROR

Development Error Tracer Disabled.

Table 14: Define IWD_CFG_DEBUG_ERROR Description

Name	IWD_CFG_DEBUG_ERROR
Initializer	IWD_STATE_ON

3.4.1.10 Define IWD_CFG_DHCP_CLIENT_ENABLED

Support DHCP (Dynamic Host Configuration Protocol) Client.

Table 15: Define IWD_CFG_DHCP_CLIENT_ENABLED Description

Name	IWD_CFG_DHCP_CLIENT_ENABLED
Initializer	TRUE

3.4.1.11 Define IWD_CFG_DHCP_HOSTNAME_LEN

Length of DHCP Hostname Option.

Table 16: Define IWD_CFG_DHCP_HOSTNAME_LEN Description

Name	IWD_CFG_DHCP_HOSTNAME_LEN
Initializer	0x08

3.4.1.12 Define IWD_CFG_DHCP_HOSTNAME_OPT_ENABLED

When is TRUE the DHCP Hostname Option is added to DHCP message.

Details :

DHCP Hostname

Table 17: Define IWD_CFG_DHCP_HOSTNAME_OPT_ENABLED Description

Name	IWD_CFG_DHCP_HOSTNAME_OPT_ENABLED
Initializer	TRUE

3.4.1.13 Define IWD_CFG_DHCP_HOSTNAME_VALUE

Value of DHCP Hostname Option.

Table 18: Define IWD_CFG_DHCP_HOSTNAME_VALUE Description

Name	IWD_CFG_DHCP_HOSTNAME_VALUE
------	-----------------------------

Initializer	"Bolero3M"
-------------	------------

3.4.1.14 Define IWD_CFG_DHCP_LEASE_OPT_ENABLED

When is TRUE the IP address lease time Option.

Table 19: Define IWD_CFG_DHCP_LEASE_OPT_ENABLED Description

Name	IWD_CFG_DHCP_LEASE_OPT_ENABLED
Initializer	FALSE

3.4.1.15 Define IWD_CFG_DHCP_MAX_MSG_SIZE_OPT_ENABLED

When is TRUE the Maximum DHCP Message Size Option is added to DHCP message.

Table 20: Define IWD_CFG_DHCP_MAX_MSG_SIZE_OPT_ENABLED Description

Name	IWD_CFG_DHCP_MAX_MSG_SIZE_OPT_ENABLED
Initializer	TRUE

3.4.1.16 Define IWD_CFG_DHCP_MAX_MSG_SIZE_VALUE

Maximum DHCP Message Size Option value.

Table 21: Define IWD_CFG_DHCP_MAX_MSG_SIZE_VALUE Description

Name	IWD_CFG_DHCP_MAX_MSG_SIZE_VALUE
Initializer	(iwui16)0x05dc

3.4.1.17 Define IWD_CFG_DHCP_PARAMETER_REQ_LIST_LEN

This is the Length of Parameter Request List. User shall add the list of parameters in IpWay_Cfg.cfile in the IWD_CFG_Dhcp_ParameterReqListarray.

Table 22: Define IWD_CFG_DHCP_PARAMETER_REQ_LIST_LEN Description

Name	IWD_CFG_DHCP_PARAMETER_REQ_LIST_LEN
Initializer	(iwui8)0x04

3.4.1.18 Define IWD_CFG_ICMP_ENABLED

Support ICMP (Internet Control Message Protocol).

Table 23: Define IWD_CFG_ICMP_ENABLED Description

Name	IWD_CFG_ICMP_ENABLED
Initializer	TRUE

3.4.1.19 Define IWD_CFG_ICMP_TTL

Default Time-to-live value for ICMP packets.

Table 24: Define IWD_CFG_ICMP_TTL Description

Name	IWD_CFG_ICMP_TTL
Initializer	(iwui8)0x80

3.4.1.20 Define IWD_CFG_INET6

IPv6 Enables (TRUE) or Disables (FALSE).

Table 25: Define IWD_CFG_INET6 Description

Name	IWD_CFG_INET6
Initializer	FALSE

3.4.1.21 Define IWD_CFG_MAX_NETID_SUPPORTED

Maximum number of Network interface (Ethernet controllers) supported.

Table 26: Define IWD_CFG_MAX_NETID_SUPPORTED Description

Name	IWD_CFG_MAX_NETID_SUPPORTED
Initializer	1

3.4.1.22 Define IWD_CFG_PATHMTU_DISCOVERY_SUPPORT

Support the discovery of the maximum transmission unit on a path according to IETF RfC 1191.

Table 27: Define IWD_CFG_PATHMTU_DISCOVERY_SUPPORT Description

Name	IWD_CFG_PATHMTU_DISCOVERY_SUPPORT
Initializer	TRUE

3.4.1.23 Define IWD_CFG_SCHEDULER_PERIOD_MS

This define is not used in IpWay but is useful to help the user to choose the right period of `IWF_Scheduler()` ;.

Details :

Considering that in the worst case the number of clock cycles needed to execute this function is approximately 96000 ticks then the Minimal Scheduler Period (MSP) will be $MSP = Round((96000 / \text{System Clock Frequency [Hz]} \times 1000) \text{ [ms]})$, where 'Round' means round to biggest integer. Let's do an example: Supposing that the system clock is 64MHz, then $MSP = Round((96000 / 64000000) \times 1000) \text{ [ms]} = 2$;

Table 28: Define IWD_CFG_SCHEDULER_PERIOD_MS Description

Name	IWD_CFG_SCHEDULER_PERIOD_MS
Initializer	100

3.4.1.24 Define IWD_CFG_SOCKET_MAX

Maximum number of sockets (TCP+UDP).

Table 29: Define IWD_CFG_SOCKET_MAX Description

Name	IWD_CFG_SOCKET_MAX
Initializer	IWD_CFG_TCP_SOCKET_MAX+IWD_CFG_UDP_SOCKET_MAX

3.4.1.25 Define IWD_CFG_STATIC_IP_ADDR_GATEWAY_0

Static Ip Address of Gateway related to Network Interface 0.

Table 30: Define IWD_CFG_STATIC_IP_ADDR_GATEWAY_0 Description

Name	IWD_CFG_STATIC_IP_ADDR_GATEWAY_0
Initializer	IWM_IP_ADDR(192,168,0,100)

3.4.1.26 Define IWD_CFG_STATIC_SUBNET_MASK_NETIF_0

Static Subnet Mask of Network Interface 0.

Table 31: Define IWD_CFG_STATIC_SUBNET_MASK_NETIF_0 Description

Name	IWD_CFG_STATIC_SUBNET_MASK_NETIF_0
Initializer	24

3.4.1.27 Define IWD_CFG_TCP_ENABLED

Support of TCP (Transmission Control Protocol).

Table 32: Define IWD_CFG_TCP_ENABLED Description

Name	IWD_CFG_TCP_ENABLED
Initializer	TRUE

3.4.1.28 Define IWD_CFG_TCP_SOCKET_MAX

Maximum number of TCP sockets.

Table 33: Define IWD_CFG_TCP_SOCKET_MAX Description

Name	IWD_CFG_TCP_SOCKET_MAX
Initializer	(iwui16)0x000a

3.4.1.29 Define IWD_CFG_UDP_ENABLED

Support of UDP (User Data Protocol).

Table 34: Define IWD_CFG_UDP_ENABLED Description

Name	IWD_CFG_UDP_ENABLED
Initializer	TRUE

3.4.1.30 Define IWD_CFG_UDP_SOCKET_MAX

Maximum number of UDP sockets.

Table 35: Define IWD_CFG_UDP_SOCKET_MAX Description

Name	IWD_CFG_UDP_SOCKET_MAX
Initializer	(iwui16)0x000a

3.4.1.31 Define IWD_IP_ADDR_ASSIGN_MTD_AUTOIP_PRIO

Priority of AUTOIP Ip Address Assignment method.

Table 36: Define IWD_IP_ADDR_ASSIGN_MTD_AUTOIP_PRIO Description

Name	IWD_IP_ADDR_ASSIGN_MTD_AUTOIP_PRIO
Initializer	3

3.4.1.32 Define IWD_IP_ADDR_ASSIGN_MTD_STATIC_PRIO

Priority of STATIC Ip Address Assignment method.

Table 37: Define IWD_IP_ADDR_ASSIGN_MTD_STATIC_PRIO Description

Name	IWD_IP_ADDR_ASSIGN_MTD_STATIC_PRIO
Initializer	2

3.4.1.33 Define IWM_TIMER_GET_TIME

Get the time instant of the IpWay internal watch.

Table 38: Define IWM_TIMER_GET_TIME Description

Name	IWM_TIMER_GET_TIME
Initializer	IWG_Timer_LocalTime

3.4.2 Macros Reference

Macros supported by the IpWay library.

3.4.2.1 Macro IWM_IP_ADDR

Details :

ip address macro

Prototype : IWM_IP_ADDR(msb,mlsb,lmsb,lsb)

3.4.2.2 Macro IWM_CHECKSUM_FOLD_32BIT_SUM_TO_16BIT

Prototype : IWM_CHECKSUM_FOLD_32BIT_SUM_TO_16BIT(sum)

3.4.2.3 Macro IWM_CPU_H2NL

Converts a 32bit integer from Host byte order to Network byte order.

Prototype : IWM_CPU_H2NL(Long)

3.4.2.4 Macro IWM_CPU_H2NS

Converts a 16bit integer from Host byte order to Network byte order.

Prototype : IWM_CPU_H2NS(Short)

3.4.2.5 Macro IWM_CPU_N2HL

Converts a 32bit integer from Network byte order to Host byte order.

Prototype : IWM_CPU_N2HL(Long)

3.4.2.6 Macro IWM_CPU_N2HS

Converts a 16bit integer from Network byte order to Host byte order.

Prototype : IWM_CPU_N2HS(Short)

3.4.2.7 Macro IWM_DEBUG_CHECK

Prototype : IWM_DEBUG_CHECK(Condition,ServiceID)

3.4.2.8 Macro IWM_IP_ADDR_IS_BROADCAST

Prototype : IWM_IP_ADDR_IS_BROADCAST(ip)

3.4.2.9 Macro IWM_IP_ADDR_IS_MULTICAST

Prototype : IWM_IP_ADDR_IS_MULTICAST(ip)

3.4.2.10 Macro IWM_IP_CLASS_A

Prototype : IWM_IP_CLASS_A(ip)

3.4.2.11 Macro IWM_IP_CLASS_B

Prototype : IWM_IP_CLASS_B(ip)

3.4.2.12 Macro IWM_IP_CLASS_C

Prototype : IWM_IP_CLASS_C(ip)

3.4.2.13 Macro IWM_IP_CLASS_D

Prototype : IWM_IP_CLASS_D(ip)

3.4.2.14 Macro IWM_IP_GET_HEADER_LEN

Prototype : IWM_IP_GET_HEADER_LEN(ip_header_ptr)

3.4.2.15 Macro IWM_IP_GET_IP_VERSION

Prototype : IWM_IP_GET_IP_VERSION(ip_header_ptr)

3.4.2.16 Macro IWM_MEM_SIZEOF

Prototype : IWM_MEM_SIZEOF(type)

3.4.2.17 Macro IWM_MTX_NETID_LOCK

Details :

Lock a service not reentrant for the same network interface, referred by NetId, if it is not locked otherwise return an error condition.

Prototype : IWM_MTX_NETID_LOCK(RetVal,ServiceId,NetId)

Table 39: IWM_MTX_NETID_LOCK Arguments

Name	Direction	Description
ServiceId	input	Service identifier in the context of mutex of functions not reentrant for the same network interface.
NetId	input	Network interface identifier.
RetVal	input	Return value in case of mutex already locked.

3.4.2.18 Macro IWM_MTX_NETID_UNLOCK

Details :

Unlock a service not reentrant for the same network interface, referred by NetId. if it is not locked otherwise return an error condition.

Prototype : IWM_MTX_NETID_UNLOCK(ServiceId, NetId)

Table 40: IWM_MTX_NETID_UNLOCK Arguments

Name	Direction	Description
ServiceId	input	Service identifier in the context of mutex of functions not reentrant for the same network interface.
NetId	input	Network interface identifier.

3.4.2.19 Macro IWM_MTX_SCKID_LOCK

Details :

Lock a service not reentrant for the same socket, referred by ServiceId, if it is not locked otherwise return an error condition.

Prototype : IWM_MTX_SCKID_LOCK(RetVal, ServiceId, SocketId)

Table 41: IWM_MTX_SCKID_LOCK Arguments

Name	Direction	Description
ServiceId	input	Service identifier in the context of mu-

Name	Direction	Description
		tex of functions not reentrant for the same socket.
SocketId	input	Socket identifier.
RetVal	input	Retun value in case of mutex al-ready locked.

3.4.2.20 Macro IWM_MTX_SCKID_UNLOCK

Details :

Unlock the mutex of a service not reentrant for the same socket, referred by ServiceId.

Prototype : IWM_MTX_SCKID_UNLOCK(ServiceId, SocketId)

Table 42: IWM_MTX_SCKID_UNLOCK Arguments

Name	Direction	Description
ServiceId	input	Service identifier in the contaxt of mu-tex of functions not reentrant for the same socket.
SocketId	input	Socket identifier.

3.4.2.21 Macro IWM_MTX_SID_LOCK

Details :

Lock a service referred by ServiceId if the mutex associated is not locked otherwise return an error condition.

Prototype : IWM_MTX_SID_LOCK(RetVal, ServiceId)

Table 43: IWM_MTX_SID_LOCK Arguments

Name	Direction	Description
ServiceId	input	Service identifier in the context of mu-tex of functions on-ly.
RetVal	input	Retun value in case of mutex al-ready locked.

3.4.2.22 Macro IWM_MTX_SID_UNLOCK

Details :

Unlock the mutex associated to the service referred by ServiceId.

Prototype : IWM_MTX_SID_UNLOCK(ServiceId)

Table 44: IWM_MTX_SID_UNLOCK Arguments

Name	Direction	Description
ServiceId	input	Service identifier in the context of mutex.

3.4.3 Enum Reference

Enumeration of all constants supported by the IpWay library.

3.4.3.1 Enumeration IWD_IpAddrAssignMtdType

Specification of IP address assignment policy.

Table 45: Enumeration IWD_IpAddrAssignMtdType Values

Name	Initializer	Description
IWD_IP_ADDR_ASSIGN_MTD_AUTOIP	0	Auto-IP address assignment method (timing according to IETF RFC 3927).
IWD_IP_ADDR_ASSIGN_MTD_AUTOIP_DOIP		Auto-IP address assignment method (timing according to ISO 13400 (DoIP)).
IWD_IP_ADDR_ASSIGN_MTD_DHCP		DHCP Ip address assignment method.
IWD_IP_ADDR_ASSIGN_MTD_STATIC		STATIC Ip address assignment method.
IWD_IP_ADDR_ASSIGN_MTD_NULL		Ip address assignment method NOT AASIGNED.

3.4.3.2 Enumeration IWD_IpAddrType

Ip address type, Multicast and Unicast.

Table 46: Enumeration IWD_IpAddrType Values

Name	Initializer	Description
IWD_MULTICAST	0	Identifier of Multicast Ip address.
IWD_UNICAST		Identifier of Unicast Ip address.

3.4.3.3 Enumeration IWD_NetStateType

Network Interface State type.

Table 47: Enumeration IWD_NetStateType Values

Name	Initializer	Description
IWD_NET_STATE_ONLINE	0	Network Interface Online - stable state.
IWD_NET_STATE_FROZEN		Network Interface Frozen - stable state.
IWD_NET_STATE_OFFLINE		Network Interface Offline - stable state.
IWD_NET_STATE_STARTUP		Network Interface in Start-up state - NOT stable state.
IWD_NET_STATE_SHUTDOWN		Network Interface in Shutdown state - NOT stable state.

3.4.3.4 Enumeration IWD_ProtocolType

Protocol type used by a socket.

Table 48: Enumeration IWD_ProtocolType Values

Name	Initializer	Description
IWD_PROTOCOL_ICMP	0x01	
IWD_PROTOCOL_TCP	0x06	
IWD_PROTOCOL_UDP	0x11	

3.4.3.5 Enumeration IWD_TcpCopyType

Tcp Copy Type.

Table 49: Enumeration IWD_TcpCopyType Values

Name	Initializer	Description
IWD_TCP_COPY	0	TCP Copyless Disabled.
IWD_TCP_COPYLESS		TCP Copyless Enabled.

3.4.4 Function Reference

Functions supported by the IpWay library.

3.4.4.1 Function EIF_Cbk

Callback from Ethernet LLD.

Figure 2: Function EIF_Cbk References.



Details :

Once that the Ethernet Low Level Driver (LLD) receives a new frame it shall be forwarded to high levels by calling this service. This service forwards the received frame to IpWay.

Binding info: nothing to do.

Prototype : void EIF_Cbk(IWD_NetIdType NetId, EID_RxBufferType *RxBufferPtr);

Table 50: EIF_Cbk Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Index of the Ethernet controller within the context of IpWay.
EID_RxBufferType*	RxBufferPtr	input	Pointer to general purpose Rx Buffer structure.

3.4.4.2 Function EIF_ChangeState

Change the controller state.

Details :

The service `EIF_ChangeState()` changes the state of ethernet controller identified by `NetId`. This service is NOT invoked by IpWay but shall be invoked by the user at application level; have look to the examples in the package.

Binding info: In order to change the state of controller this service shall call a low level service of Ethernet driver able to change the state of the Ethernet controller.

The parameter `CtrlState` indicates the state of Ethernet controller within the context of IpWay. For each state defined in `EID_CtrlStateType` shall be called the relative service of Ethernet driver; consider that the `EID_CtrlStateType` defines standard items but the user can modify it according his needs.

 **Note:** Is mandatory calling the service `EIF_Init()` before calling this service.

Return : Error status.

Prototype : `IWD_EndType EIF_ChangeState(IWD_NetIdType NetId, EID_CtrlStateType CtrlState);`

Table 51: EIF_ChangeState Arguments

Type	Name	Direction	Description
<code>IWD_NetIdType</code>	<code>NetId</code>	input	Index of the Ethernet controller within the context of IpWay.
<code>EID_CtrlStateType</code>	<code>CtrlState</code>	input	State of controller wanted.

Table 52: EIF_ChangeState Return Values

Name	Description
<code>IWD_E_OK</code>	Request accepted.
<code>IWD_E_NOT_OK</code>	Request not accepted.

3.4.4.3 Function EIF_CtrlInit

Initializes ethernet controller.

Details :

Binding info: In order to initialize the Ethernet controller the service `EIF_CtrlInit()` shall call a low level service of Ethernet driver able to initialize the Ethernet controller.

The parameter `NetId` is the identifier of Ethernet controller within the context of IpWay. The user shall configure, in `EthernetIf_Cfg.h` configuration file, the Ethernet controller identifier within the context of Ethernet driver;

Return : Error status.

Prototype : `IWD_EndType EIF_CtrlInit(IWD_NetIdType NetId);`

Table 53: EIF_CtrlInit Arguments

Type	Name	Direction	Description
<code>IWD_NetIdType</code>	<code>NetId</code>	input	Index of the Ethernet controller within the context of IpWay.

Table 54: EIF_CtrlInit Return Values

Name	Description
IWD_E_OK	Controller initialized.
IWD_E_NOT_OK	Controller NOT initialized.

3.4.4.4 Function EIF_GetLinkStatus

Get the link status of Ethernet Interface.

Details :

The service EIF_GetLinkStatus() gets the link status of ethernet interface referred by NetId.

This service is used by IpWay in order to understand if suddenly, while the network state is ONLINE, the physical link becomes offline (e.g. user disconnect the Ethernet Cable).

Return : State of Ethernet Interface.

Prototype : EID_CtrlLinkStatusType EIF_GetLinkStatus(IWD_NetIdType NetId);

Table 55: EIF_GetLinkStatus Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Network interface identifier which match with Ethernet interface controller.

3.4.4.5 Function EIF_GetMACAddr

Get MAC Address.

Details :

Binding info: In order to retrieve the MAC address the service EIF_GetMACAddr() shall call a low level service of Ethernet driver ables to get the MAC address of the Ethernet controller.

 **Note:** Is mandatory calling the service EIF_Init() before calling this service.

Prototype : void EIF_GetMACAddr(IWD_NetIdType NetId, iwui8 *MACAddrPtr);

Table 56: EIF_GetMACAddr Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Index of the Ethernet controller within the context of IpWay.
iwui8*	MACAddrPtr	output	Pointer to MAC address stored in network byte order.

3.4.4.6 Function EIF_GetState

Get state of Ethernet Interface.

Details :

The service EIF_GetState() gets the state of ethernet interface referred by NetId.

Return : State of Ethernet Interface.

Prototype : EID_CtrlStateType EIF_GetState(IWD_NetIdType NetId);

Table 57: EIF_GetState Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Network interface identifier which match with Ethernet interface controller.

3.4.4.7 Function EIF_GetTxBuffer

Get Tx Buffer to prepare the transmission.

Figure 3: Inverse references of fuction EIF_GetTxBuffer



Details :

Binding info: In order to get the transmission buffer the service EIF_GetTxBuffer() shall call a low level service of Ethernet driver ables to prepare a Tx Buffer that will be used by IpWay to load the Frame to send.

Mandatory: this service must fill some fields of structure pointed by TxBufferPtr in particular:

BufPtr: pointer to memory buffer allocated by Ethernet driver

BufLength: length of memory buffer

IpWay/RT uses this service together with the EIF_TxBuffer() service.

Note: Is mandatory calling the service EIF_Init() before calling this service.

Return : Error status.

Prototype : IWD_EndType EIF_GetTxBuffer(IWD_NetIdType NetId, EID_TxBufferType *TxBufferPtr);

Table 58: EIF_GetTxBuffer Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Index of the Ethernet controller within the context of IpWay.

Type	Name	Direction	Description
EID_TxBufferType*	TxBufferPtr	input, output	Pointer to general purpose Tx Buffer structure.

Table 59: EIF_GetTxBuffer Return Values

Name	Description
IWD_E_OK	Request accepted.
IWD_E_NOT_OK	Request not accepted.

3.4.4.8 Function EIF_Init

Initializes the Ethernet Interface.

Details :

This service initializes internal variables of Ethernet interface

Binding info: User could add here the call to the Ethernet Driver initialization function.

 **Note:** Is mandatory call this service before starting the IpWay and the other ethernet interface functions.

Prototype : void EIF_Init(const EID_CfgType *CfgPtr);

Table 60: EIF_Init Arguments

Type	Name	Direction	Description
constEID_CfgType*	CfgPtr	input	Pointer to configuration structure.

3.4.4.9 Function EIF_TxBuffer

Send frame stored in Tx Buffer.

Figure 4: Inverse references of fuction EIF_TxBuffer



Details :

The service EIF_TxBuffer() sends the frame contained in Tx buffer prepared byEIF_GetTxBuffer.

Binding info: In order to send a frame this servise shall call a low level service of Ethernet driver able to send a frame.

IpWay/RT uses this service together with the EIF_GetTxBuffer() service.

 **Note:** Is mandatory calling the service EIF_Init() before calling this service.Is mandatory calling the service EIF_GetTxBuffer() before calling this service.

Return : Error status.

Prototype : IWD_EndType EIF_TxBuffer(IWD_NetIdType NetId,
EID_TxBufferType *TxBufferPtr);

Table 61: EIF_TxBuffer Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Index of the Ethernet controller within the context of IpWay.
EID_TxBufferType*	TxBufferPtr	input	Pointer to general purpose Tx Buffer structure.

Table 62: EIF_TxBuffer Return Values

Name	Description
IWD_E_OK	Request accepted.
IWD_E_NOT_OK	Request not accepted.

3.4.4.10 Function IWF_BindSocket

Bind a socket to a local resource.

Details :

The service IWF_BindSocket() binds the socket specified by parameter SocketId to ip address and port specified by parameters NetId and PortPtr respectively.

In order to bind the socket to any Ip Address, the IPAddr param specified by NetId must be set to "IWD_NETID_ANY".

Return : Result of the 'bind socket' request; in case of PortPtr parameter specified as IWD_PORT_ANY, the function return the Port field of socket as function parameter.

 **Note**: Reentrant only for different Sockets (SocketId), Synchronous.

Prototype : IWD_EndType IWF_BindSocket(IWD_SocketIdType SocketId,
IWD_NetIdType NetId, iwui16 *PortPtr);

Table 63: IWF_BindSocket Arguments

Type	Name	Direction	Description
IWD_SocketIdType	SocketId	input	Socket identifier.
IWD_NetIdType	NetId	input	Network identifier representing the local IP address and Ethernet controller to bind to the socket.
iwui16*	PortPtr	input, output	Pointer to local port to which the socket shall be bound. If this param is specified as IWD_PORT_ANY, the IpWay

Type	Name	Direction	Description
			stack shall choose the local port automatically from 49152 to 65535 and shall update the parameter to the chosen value. Note: The range from 49152 to 65535 is used for custom or temporary purposes.

Table 64: IWF_BindSocket Return Values

Name	Description
IWD_E_OK	Request accepted: Socket bound (PortPtr VALID).
IWD_E_NOT_OK	Request not accepted: Socket NOT bound (PortPtr NOT VALID).

3.4.4.11 Function IWF_ChangeState

Change state of Network Interface.

Details :

The service `IWF_ChangeState()` changes the state of network interface referred by `NetId`.

If `IWD_NET_STATE_ONLINE` is requested: IpWay starts the acquisition of Ip Address according to the Ip address assignment policy configured.

If `IWD_NET_STATE_OFFLINE` is requested: IpWay stops the activity on the interface by removing all resources allocated (e.g close all sockets, and so on). using the specified EthIf controller).

If `IWD_NET_STATE_FROZEN` is requested: IpWay froze all the activities on the interface. User shall re-call this service with state `IWD_NET_STATE_ONLINE` to recover the activities.

Return : Result of 'change state' request.

 **Note:** Reentrant only for different Network interfaces (`NetId`), Asynchronous.

Prototype : `IWD_EndType IWF_ChangeState(IWD_NetIdType NetId, IWD_NetStateType State);`

Table 65: IWF_ChangeState Arguments

Type	Name	Direction	Description
<code>IWD_NetIdType</code>	<code>NetId</code>	input	Network interface identifier which match with Ethernet interface controller.
<code>IWD_NetStateType</code>	<code>State</code>	input	Requested state.

Table 66: IWF_ChangeState Return Values

Name	Description
IWD_E_OK	Request accepted.
IWD_E_NOT_OK	Request not accepted.

3.4.4.12 Function IWF_CloseSocket

Close a Socket.

Details :

The service IWF_CloseSocket () closes the socket specified by SocketId.

In case of UDP socket, the socket and all related resources are immediately released.

In case of TCP socket, the socket is terminated according to the procedure described in RFC793, in particular:

case CloseForced = TRUE matches with the ABORT Call

case CloseForced = FALSE matches with the CLOSE Call.

Return : Result of the 'close socket' request.

 **Note:** Reentrant only for different Sockets (SocketId), Asynchronous.

Prototype : IWD_EndType IWF_CloseSocket(IWD_SocketIdType SocketId, iwbool CloseForced);

Table 67: IWF_CloseSocket Arguments

Type	Name	Direction	Description
IWD_SocketIdType	SocketId	input	Socket identifier.
iwbool	CloseForced	input	Indicates the closing method.

Table 68: IWF_CloseSocket Return Values

Name	Description
IWD_E_OK	Request accepted.
IWD_E_NOT_OK	Request not accepted.

3.4.4.13 Function IWF_CreateSocket

Create a Socket.

Details :

IWF_CreateSocket () allocate a new socket.

Return : Result of the 'create socket' request.

 **Note:** Not Reentrant, Synchronous.

Prototype : IWD_EndType IWF_CreateSocket(IWD_NetProtType NetProt, IWD_ProtocolType Protocol, IWD_SocketIdType *SocketIdPtr, iwui16 MaxConn);

Table 69: IWF_CreateSocket Arguments

Type	Name	Direction	Description
IWD_NetProtType	NetProt	input	Network protocol family.
IWD_ProtocolType	Protocol	input	Transport protocol family.
IWD_SocketIdType*	SocketIdPtr	output	Pointer to socket identifier representing the requested socket.
iwui16	MaxConn	input	Max number of TCP connections that can be active contemporarily.

Table 70: IWF_CreateSocket Return Values

Name	Description
IWD_E_OK	Request accepted: SocketIdPtr VALID.
IWD_E_NOT_OK	Request not accepted: SocketIdPtr NOT VALID.

3.4.4.14 Function IWF_DhcpAddOption

Add DHCP Options.

Figure 5: Inverse references of function IWF_DhcpAddOption



Details :

The service IWF_DhcpAddOption() adds the dhcp option to dhcp message.

A set of options could be added by default at configuration time, in particular look for IWD_CFG_Dhcp_ParameterReqList[IWD_CFG_DHCP_PARAMETER_REQ_LIST_LEN] in IpWay_Cfg.c and IpWay_Cfg.h files.

Return : Result of the request of adding of a DHCP option.

 **Note:** Reentrant only for different Network interfaces (NetId), Synchronous.

Prototype : IWD_EndType IWF_DhcpAddOption(IWD_NetIdType NetId, iwui8 Option, const iwui8 *OptionPtr, iwui8 OptionLength);

Table 71: IWF_DhcpAddOption Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Network interface identifier.

Type	Name	Direction	Description
iwui8	Option	input	DHCP option.
constiwui8*	OptionPtr	input	Pointer to DHCP option data.
iwui8	OptionLength	input	Length of DHCP option data.

Table 72: IWF_DhcpAddOption Return Values

Name	Description
IWD_E_OK	Option added successfully.
IWD_E_NOT_OK	Option NOT added.

3.4.4.15 Function IWF_GetIpConfig

Get IP configuration.

Details :

The service `IWF_GetIpConfig()` retrieves the Ip configuration (IP addresses and subnet mask) of Network interface identified by NetId.

Return : Result of 'get configuration' request; the IP configuration is returned as function parameters.

 **Note:** Reentrant, Synchronous.

Prototype : `IWD_EndType IWF_GetIpConfig(IWD_NetIdType NetId, iwui32 *IPAddrPtr, iwui8 *SubnetMaskPtr, iwui32 *GatewayIPAddrPtr);`

Table 73: IWF_GetIpConfig Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Network interface identifier for which the IP configuration shall be retrieved.
iwui32*	IPAddrPtr	output	Pointer to IP address retrieved.
iwui8*	SubnetMaskPtr	output	Pointer to Subnet Mask retrieved.
iwui32*	GatewayIPAddrPtr	output	Pointer to Gateway Ip address retrieved.

Table 74: IWF_GetIpConfig Return Values

Name	Description
IWD_E_OK	Parameters retrieved successfully.
IWD_E_NOT_OK	Parameters NOT retrieved.

3.4.4.16 Function IWF_GetMACAddr

Get MAC Address.

Details :

The service IWF_GetMACAddr() :

If IPAddrPtr is equal to IWD_NULL_PTR, retrieves the MAC address of ethernet interface controller specified by NetId.

If IPAddrPtr is different by IWD_NULL_PTR, looks for the MAC address in the ARP table of network interface identified by NetId, by entry IPAddrPtr, If no MAC is retrieved an ARP request is sent.

Return : Result of 'get MAC address' request; the MAC address is returned as function parameter.

 **Note:** Reentrant only for different Network interfaces (NetId), Asynchronous.

Prototype : IWD_EndType IWF_GetMACAddr(IWD_NetIdType NetId, iwui32 *IPAddrPtr, iwui8 *MACAddrPtr);

Table 75: IWF_GetMACAddr Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Network interface identifier for which the MAC address shall be retrieved.
iwui32*	IPAddrPtr	input	Pointer to IP address.
iwui8*	MACAddrPtr	output	Pointer to MAC address stored in network byte order.

Table 76: IWF_GetMACAddr Return Values

Name	Description
IWD_E_OK	MAC address retrieved successfully.
IWD_E_NOT_OK	Request failed.

3.4.4.17 Function IWF_GetState

Get state of Network Interface.

Details :

The service IWF_GetState() gets the state of network interface referred by NetId.

Return : State of Network Interface.

Prototype : IWD_NetStateType IWF_GetState(IWD_NetIdType NetId);

Table 77: IWF_GetState Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	Network interface identifier which match with Ethernet interface controller.

3.4.4.18 Function IWF_IcmpSend

Sends an ICMP message.

Details :

The service IWF_IcmpSend() sends a ICMP message to destination socket specified by DestSocketPtr.

Return : Result of 'ICMP send' request.

 **Note:** Reentrant only for different Network interfaces (NetId), Synchronous.

Prototype : IWD_EndType IWF_IcmpSend(IWD_NetIdType NetId, IWD_SocketType *DestSocketPtr, iwui8 TypeOfService, iwui8 TimeToLive, iwui8 Type, iwui8 Code, iwui8 *PayloadPtr, iwui16 PayloadLength);

Table 78: IWF_IcmpSend Arguments

Type	Name	Direction	Description
IWD_NetIdType	NetId	input	IP address identifier representing the local IP address and EthIf controller which shall be used for transmission of the ICMP message.
IWD_NetIdType	NetId	input	Network interface identifier; Identify the source IP Address.
IWD_SocketType*	DestSocketPtr	input	Pointer to destination socket; Identify the IP Address and port of destination host.
iwui8	TypeOfService	input	Type of service.
iwui8	TimeToLive	input	Time to live value to be used for the ICMP message. If 0 is specified the default value shall be used.
iwui8	Type	input	Type of ICMP message.
iwui8	Code	input	Code of ICMP message.
iwui8*	PayloadPtr	input	Pointer to data of ICMP message.
iwui16	PayloadLength	input	Length of ICMP message.

Table 79: IWF_IcmpSend Return Values

Name	Description
IWD_E_OK	The ICMP message sent successfully.
IWD_E_NOT_OK	The ICMP message not sent.

3.4.4.19 Function IWF_Init

This service initializes the IPWAY TCP/IP Stack.

Details :

IWF_Init() initializes all services enabled in the configuration file IpWay_Cfg.h.

 **Note:** Is mandatory call this service before starting the IpWay processing. Not Reentrant, Synchronous.

Return : nothing.

Prototype : void IWF_Init(const IWD_CfgType *CfgPtr);

Table 80: IWF_Init Arguments

Type	Name	Direction	Description
const IWD_CfgType*	CfgPtr	input	Pointer to the configuration data of the IpWay library.

3.4.4.20 Function IWF_Scheduler

Schedules the TCP/IP stack. (Entry point for scheduling).

Details :

The scheduler function shall be called periodically by the application.

Prototype : void IWF_Scheduler(void);

3.4.4.21 Function IWF_TcpClose

Close a TCP connection.

Details :

The service IWF_TcpClose() closes a TCP connection specified by TCBIId.

According to the procedure described in RFC793:

case CloseForced = TRUE matches with the ABORT Call

case CloseForced = FALSE matches with the CLOSE Call.

Return : Result of 'TCP close connection' request.

 **Note:** Not Reentrant, Asynchronous

Prototype : IWD_EndType IWF_TcpClose(IWD_TCBIIdType TCBIId, iwbool CloseForced);

Table 81: IWF_TcpClose Arguments

Type	Name	Direction	Description
IWD_TCBIdType	TCBId	input	Connection identifier.
iwbool	CloseForced	input	Indicates the closing method.

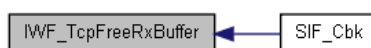
Table 82: IWF_TcpClose Return Values

Name	Description
IWD_E_OK	Request accepted.
IWD_E_NOT_OK	Request not accepted.

3.4.4.22 Function IWF_TcpFreeRxBuffer

Confirmation of data received.

Figure 6: Inverse references of fuction IWF_TcpFreeRxBuffer



Details :

The service `IWF_TcpFreeRxBuffer()` increases the TCP receive window of the connection specified by TCBId.

Return : Result of 'free rx buffer' request.

 **Note:** Not Reentrant, Synchronous

Prototype : `IWD_EndType IWF_TcpFreeRxBuffer(IWD_TCBIdType TCBId, iwui32 Bytes);`

Table 83: IWF_TcpFreeRxBuffer Arguments

Type	Name	Direction	Description
IWD_TCBIdType	TCBId	input	Connection identifier.
iwui32	Bytes	input	Number of bytes consumed by the upper layer.

Table 84: IWF_TcpFreeRxBuffer Return Values

Name	Description
IWD_E_OK	Request accepted.
IWD_E_NOT_OK	Request not accepted.

3.4.4.23 Function IWF_TcpOpen

Tcp Active Open Connection.

Details :

The service `IWF_TcpOpen()` starts a TCP connection. The connection is identified by the local socket `SocketId` and by the destination socket identified by `DestSocketPtr`.

In case of Active connection (`ActiveOpen = TRUE`) IpWay starts the 3-Way Handshake algorithm (RFC793), otherwise the socket remains in listen state waiting for connection request from remote host.

Return : Result of 'TCP open connection' request.

 **Note:** Not Reentrant, Asynchronous

Prototype : `IWD_EndType IWF_TcpOpen(IWD_SocketIdType SocketId, IWD_SocketType *DestSocketPtr, iwbool ActiveOpen);`

Table 85: IWF_TcpOpen Arguments

Type	Name	Direction	Description
IWD_SocketIdType	SocketId	input	Socket identifier of local resource.
IWD_SocketType*	DestSocketPtr	input	Pointer to destination socket; Identify the IP Address and port of destination host; this field is valid only in case of active connection.
iwbool	ActiveOpen	input	Indicates if the open connection is passive (FALSE) or active (TRUE).

Table 86: IWF_TcpOpen Return Values

Name	Description
IWD_E_OK	Request accepted: Open connection started.
IWD_E_NOT_OK	Request not accepted: Open connection FAILED.

3.4.4.24 Function IWF_TcpSend

Send TCP message.

Details :

The service `IWF_TcpSend()` sends a TCP message on the connection specified by `TCBId`.

This is an asynchronous request, it means that the message is appended to the output queue waiting to be sent.

Return : Result of 'TCP send message' request.

 **Note:** Not Reentrant, Asynchronous

Prototype : IWD_EndType IWF_TcpSend(IWD_TCBIdType TCBId, iwui8 TypeOfService, iwui8 *PayloadPtr, iwui32 PayloadLength, IWD_TcpCopyType TcpCopy);

Table 87: IWF_TcpSend Arguments

Type	Name	Direction	Description
iwui8	TypeOfService	input	Type of service.
IWD_TCBIdType	TCBId	input	Connection identifier.
iwui8*	PayloadPtr	input	Pointer to a linear buffer of a PayloadLength bytes containing the payload of TCP message to send.
iwui32	PayloadLength	input	Indicates the payload size of the TCP message.
IWD_TcpCopyType	TcpCopy	input	This parameter is very useful when the application data don't change runtime (e.g. web server stored in flash memory) or when the user can guarantee that the data remains in memory for all the time needed to transmit them and receive all acks from the remote host. This feature helps to save time and memory because avoids to copy the data from application memory to TCP memory, but the data shall be always available in order to guarantee that the TCP protocol can retrieve them if needs; e.g. case of a segment retransmission. IWD_TCP_COPYLESS: Copyless Enabled IWD_TCP_COPY: Copyless Disabled.

Table 88: IWF_TcpSend Return Values

Name	Description
IWD_E_OK	TCP message queued for the transmission.
IWD_E_NOT_OK	Request failed.

3.4.4.25 Function IWF_TimerUpdate

Update the IpWay local time variable.

Details :

This function shall be called periodically in order to update the internal watch of IpWay.

Prototype : void IWF_TimerUpdate(iwui32 UpdateInMs);

3.4.4.26 Function IWF_UdpSend

Send UDP datagram.

Figure 7: Inverse references of fuction IWF_UdpSend



Details :

The service IWF_UdpSend() sends an UDP datagram from source socket specified by SocketId to a destination socket specified by DestSocketPtr.

Return : Result of 'UDP send datagram' request.

Note: Reentrant only for different Sockets (SocketId), Synchronous.

Prototype : IWD_EndType IWF_UdpSend(IWD_SocketIdType SocketId, IWD_SocketType *DestSocketPtr, iwui8 TypeOfService, iwui8 *PayloadPtr, iwui16 PayloadLength);

Table 89: IWF_UdpSend Arguments

Type	Name	Direction	Description
iwui8	TypeOfService	input	Type of service.
IWD_SocketIdType	SocketId	input	Source Socket identifier.
IWD_SocketType*	DestSocketPtr	input	Pointer to destination socket; Identify the IP Address and port of destination host.
iwui8*	PayloadPtr	input	Pointer to a linear buffer of a PayloadLength bytes containing the payload of UDP datagram to send.
iwui16	PayloadLength	input	Indicates the payload size of the UDP datagram.

Table 90: IWF_UdpSend Return Values

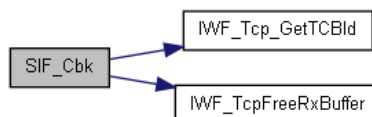
Name	Description
IWD_E_OK	UDP datagram forwarded to ethernet interface for transmission.

Name	Description
IWD_E_NOT_OK	Request failed.

3.4.4.27 Function SIF_Cbk

Callback from IpWay; receives data from transport layers.

Figure 8: Function SIF_Cbk References.



Details :

The service SIF_Cbk() gets data from transport layer (UDP or TCP).

In case of TCP socket the SocketId and DestSocketPtr parameters identify uniquely a connection. User can retrieve all information about the TCP connection inIWG_TcpTCBListarray by entering with connection identifier get by IWF_Tcp_GetTCBId() service as showed in the body of this function.

Prototype : void SIF_Cbk(IWD_SocketIdType SocketId, IWD_SocketType *DestSocketPtr, iwui8 *PayloadPtr, iwui16 PayloadLength);

Table 91: SIF_Cbk Arguments

Type	Name	Direction	Description
IWD_SocketIdType	SocketId	input	Socket identifier.
IWD_SocketType*	DestSocketPtr	input	Pointer to memory containing IP address and port of the remote host which sent the data.
iwui8*	PayloadPtr	input	Pointer to the received data.
iwui16	PayloadLength	input	Length of the received TCP segment or UDP datagram.

3.4.4.28 Function SIF_TcpConnAccepted

TCP connection request from remote host accepted.

Details :

The service SIF_TcpConnAccepted() is called by IpWay when a new request connection is established.

Prototype : void SIF_TcpConnAccepted(IWD_TCBIIdType TCBIId, IWD_SocketType *DestSocketPtr);

Table 92: SIF_TcpConnAccepted Arguments

Type	Name	Direction	Description
IWD_TCBIdType	TCBId	input	Connection identifier; this is equal to socket identifier of Ip-Way.
IWD_SocketType*	DestSocketPtr	input	Pointer to memory containing IP address and port of the remote host which sent the data.

3.4.5 Structs Reference

Data structures supported by the IpWay library.

3.4.5.1 Structure IWD_IpAddrAssignCfgType

Ip Address Assignment Configuration type.

Figure 9: Struct IWD_IpAddrAssignCfgType



Declaration :

```

typedef struct
{
    IWD_IpAddrAssignMtdType IpAddrAssignMtd,
    iwui8 IpAddrAssignMtdPrio
} IWD_IpAddrAssignCfgType;
  
```

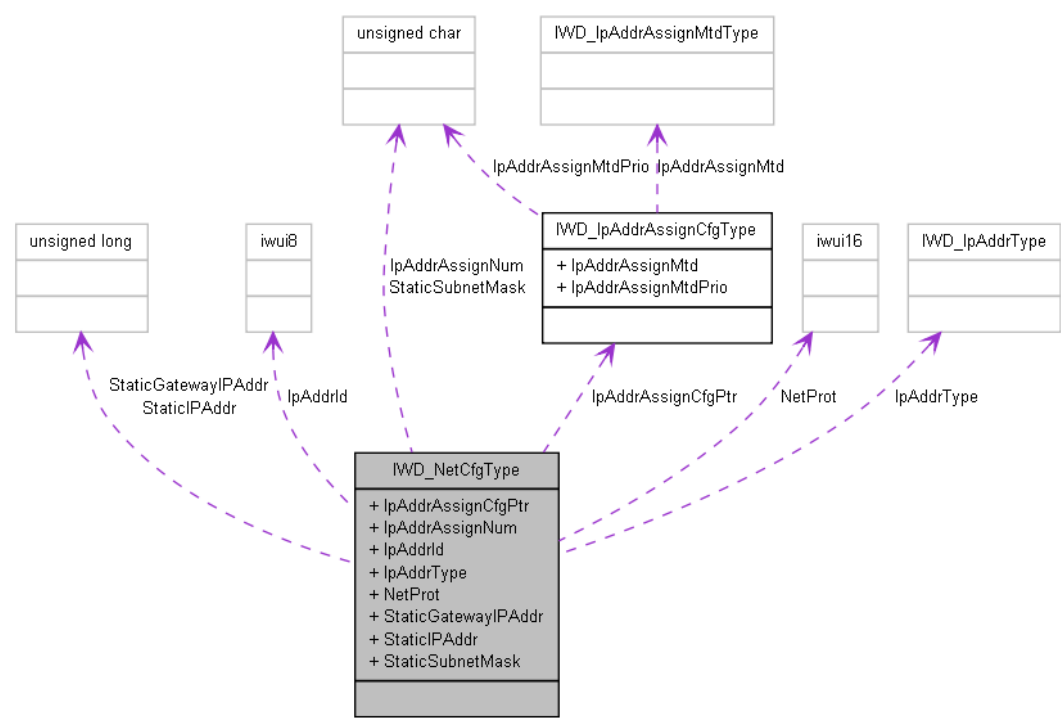
Table 93: Structure IWD_IpAddrAssignCfgType member description

Member	Description
IpAddrAssignMtd	Ip Address Assignment Method.
IpAddrAssignMtdPrio	Ip Address Assignment Method Priority.

3.4.5.2 Structure IWD_NetCfgType

Structure for Network Configuration.

Figure 10: Struct IWD_NetCfgType



Details :

The structure `IWD_NetCfgType` is a type for the external data structure containing the initialization data for the Network interface.

Declaration :

```
typedef struct
{
    const IWD_IpAddrAssignCfgType*
IpAddrAssignCfgPtr,
    iwui8 IpAddrAssignNum,
    IWD_NetIdType IpAddrId,
    IWD_IpAddrType IpAddrType,
    IWD_NetProtType NetProt,
    iwui32 StaticGatewayIPAddr[1],
    iwui32 StaticIPAddr[1],
    iwui8 StaticSubnetMask
} IWD_NetCfgType;
```

Table 94: Structure IWD_NetCfgType member description

Member	Description
IpAddrAssignCfgPtr	Pointer to Ip Address Assignment structure.

Member	Description
IpAddrAssignNum	Number of Ip Address Assignment Methods.
IpAddrId	Ip Address identifier.
IpAddrType	Ip Address Type.
NetProt	Network Protocol family.
StaticGatewayIPAddr	IP address of gateway (router).
StaticIPAddr	Static Ip Address.
StaticSubnetMask	IP address Network mask.

3.4.5.3 Structure IWD_NetIfType

Network Interface Type.

Details :

Type of the Network Interface controller that contains the local information e.g. IP address/SubnetMask/DefaultGateway

Declaration :

```
typedef struct
{
    IWD_DhcpType* Dhcp,
    iwui8 EthIfCtrlId,
    iwui32
GatewayIPAddr[IWD_IP_IPV4_ADDRESS_LEN_WORDS],
    iwui8
GatewayMACAddr[IWD_MAC_ADDR_LENGTH],
    char * Hostname,
    iwui32
IPAddr[IWD_IP_IPV4_ADDRESS_LEN_WORDS],
    IWD_IpAddrAssignMtdType
IpAddrAssignMtd,
    iwui8 MACAddr[IWD_MAC_ADDR_LENGTH],
    iwui8 MACAddrLength,
    iwui16 PMtu,
    IWD_PMTuDiscoveryType PMtuDiscovery,
    iwui32 PMtuStep,
    iwui32 PMtuSuccIncTime,
    iwui32 PMtuTooBigTime,
    IWD_NetStateType State,
    iwui32 StateTime,
    iwui8 SubnetMask
} IWD_NetIfType;
```

Table 95: Structure IWD_NetIfType member description

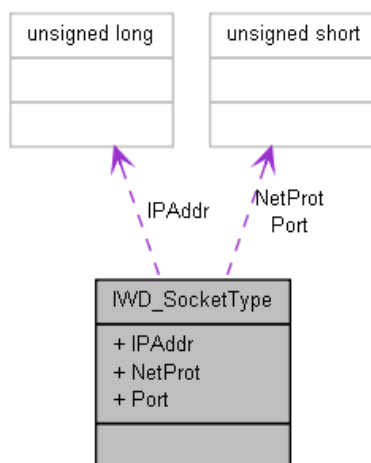
Member	Description
Dhcp	Pointer to Dhcp client element of this NetIf.
EthIfCtrlId	Ethernet Interface Controller Id.

Member	Description
GatewayIPAddr	Ip Address of Gateway.
GatewayMACAddr	MAC address of Gateway.
Hostname	Hostname of interface.
IPAddr	Ip Address of network interface.
IpAddrAssignMtd	Ip Address Assignment.
MACAddr	MAC address.
MACAddrLength	Lenght of MAC address.
PMtu	Path Mtu.
PMtuDiscovery	PMtu Discovery method used to detect the PMtu.
PMtuStep	Step size of Path Mtu Discovery alorithm.
PMtuSuccIncTime	Instant of last successful attempted increase.
PMtuTooBigTime	Instant of last Datagram Too Big message received.
State	State of network interface.
StateTime	Start Time of last change state.
SubnetMask	SubnetMask.

3.4.5.4 Structure IWD_SocketType

Socket structure which contains Ip Address, port and network protocol.

Figure 11: Struct IWD_SocketType



Declaration :

```
typedef struct
{
    iwui32 IPAddr[1],
        IWD_NetProtType NetProt,
        iwui16 Port
    } IWD_SocketType;
```

Table 96: Structure IWD_SocketType member description

Member	Description
IPAddr	
NetProt	
Port	

3.4.6 Types Reference

Types structures supported by the IpWay library.

3.4.6.1 Typedef IWD_NetIdType

Network identification type for unique identification of a network IP address and Ethernet Controller configured in the IpWay module.

Type: iwui8

3.4.6.2 Typedef IWD_NetProtType

TcpIp address families type.

Type: iwui16

3.4.6.3 Typedef IWD_SocketIdType

Socket identifier type.

Type: iwui16

3.4.7 Variables Reference

Variables supported by the IpWay library.

3.4.7.1 Variable EID_CtrlCfg

Configuration of Ethernet interface controlles.

Details :

The number of Ethernet controller is configured in IpWay_Cfg.h by IWD_CFG_MAX_NETID_SUPPORTED.

Declaration :

```
const EID_CtrlType EID_CtrlCfg[IWD_CFG_MAX_NETID_SUPPORTED]
```

3.4.7.2 Variable IWD_CFG_Dhcp_ParameterReqList

List of dhcp parameters to configure by default.

Declaration :

```
const char  
IWD_CFG_Dhcp_ParameterReqList[IWD_CFG_DHCP_PARAMETER_REQ_LIST_LEN]
```

3.5 Binding

The following sections contains information to help the user to bind the IpWay library in his environment.

3.5.1 Ethernet Interface

Interface of low level.

The Ethernet Interface shall be adapted to low level driver of Ethernet controller.

The **binding/templates/ethernet_if** folder contains the Ethernet interface file (Ethernet_If.c) that shall be modified to bind the library to the user's environment. In particular, the user shall complete the body of each function by adding specific code of low level drivers.

Here you find the list of functions that the user shall modify to adapt the library to low level driver:

- **EIF_Init:** The body of this function shall be completed by adding the call to a low level service of Ethernet driver ables to initialize the Ethernet driver.
- **EIF_CtrlInit:** The body of this function shall be completed by adding the call to a low level service of Ethernet driver ables to initialize the Ethernet controller.
- **EIF_GetMACAddr:** The body of this function shall be completed by adding the call to a low level service of Ethernet driver ables to get the MAC address of the Ethernet controller.
- **EIF_GetTxBuffer:** The body of this function shall be completed by adding the call to a low level service of Ethernet driver ables to prepare a Tx Buffer (BufPtr and BufLength) that will be used by IpWay to load the Frame to send. This function is used together with the EIF_SendFrame function.
- **EIF_SendFrame:** The body of this function shall be completed by adding the call to a low level service of Ethernet driver ables to send a frame. This function is used together with the EIF_GetTxBuffer function.
- **EIF_ChangeState:** The body of this function shall be completed by adding the call to a low level service of Ethernet driver that manages the Ethernet controller state.

To have more information about the binding of the Ethernet interface please check the functions **EIF** in the Software Specification chapter.

The **binding/examples/autosar/ethernet_if** folder contains the Ethernet interface files (Ethernet_If.c and Ethernet_If.h) ported on AUTOSAR MCAL 4.x. In case of user has an AUTOSAR MCAL 4.x could use these files for his purposes.

3.5.2 Socket Interface

Interface of application level.

The Socket Interface shall be adapted to the Application level.

The **binding/templates/socket_if** folder contains the Socket interface file (Socket_if.c) that shall be modified to bind the library to the user's environment. In particular, the user shall complete the body of each function by adding specific code of Application.

Here you find the list of functions that the user shall modify to adapt the library to application level:

- **SIF_Cbk**: The body of this function shall be completed by adding the calls to a application functions.
- **SIF_TcpConnAccepted**: The body of this function shall be completed by adding the calls to a application functions.

The release package contains an exemple in order to help the user in the binding activity.

To have more infomation about the binding of the Socket interface please check the functions **SIF** in the Software Specification chapter.

3.5.3 Timers and Scheduler

Timer and schedule period setting.

The IpWay library needs of setting of a timer to schedule the several tasks. In particular the application should call the functions IWF_Timer_Update() and IWF_Scheduler() periodically, have look to them.

4 Applications

User can find the application files in the **demios** folder of the release package. SECTION TO BE COMPLETED...

4.1 Web Server

Web server...to be completed!

5 Disclaimer

Please Read Carefully

IPWAY - Copyright 2012-2015 Giovanni Di Martino.

IPWAY is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

IPWAY is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

<http://sourceforge.net/projects/ipway/>