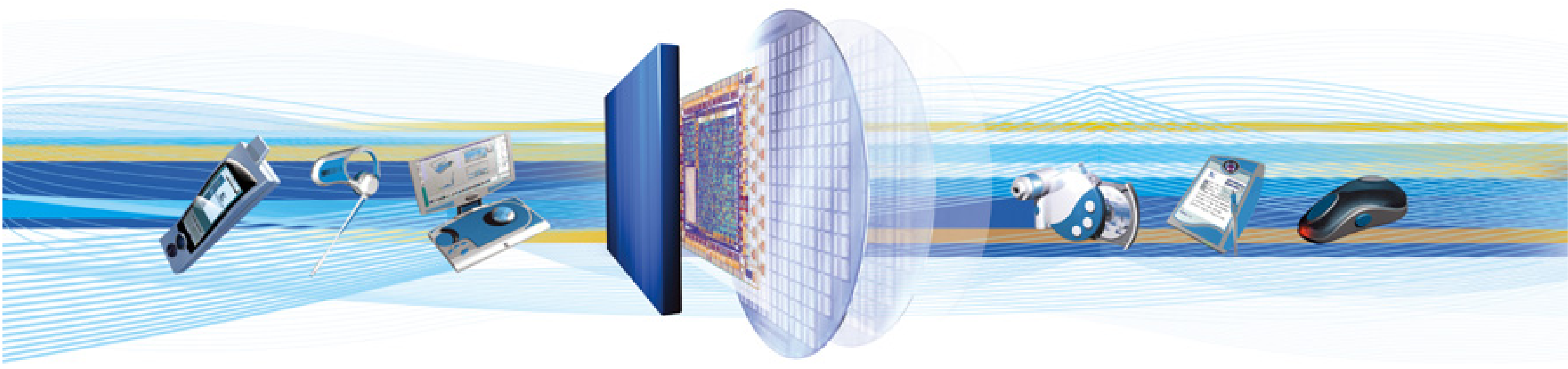
CSR Synergy®

CSR Synergy Framework 3.1.0

Ether

API Description

August 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the API between an arbitrary IP stack and CSR Router running in the CSR Scheduler. The API is called CSR IP Ether as it describes the interface in bottom of the TCP/IP stack (Network Interface layer in the TCP/IP model). This document only describes the API; the implementation is dependant on the IP stack. For an overview of the CSR IP architecture, please refer to [SYN-FRW-IP-GU].

2 Description

This section will briefly describe the purpose of introducing the CSR IP Ether API. After this section, the reader should be familiar with the location of CSR IP Ether API in the overall architecture and the reason for introducing the API.

2.1 Introduction

The CSR IP Ether API makes it easy to replace the TCP/IP stack provided in the CSR Framework release by any TCP/IP stack of the customer’s choosing. The TCP/IP stack provided by CSR is non-blocking; however, blocking TCP/IP stacks may be used since the CSR IP Ether API contains mechanisms to deal with such a stack.

The CSR IP Ether API provides the following functionality:

- Adding communication interfaces to the bottom of the IP stack.
- Handling low level data communication to and from the IP stack.

2.2 Reference Model

The CSR IP Ether API and its location relative to the TCP/IP stack.

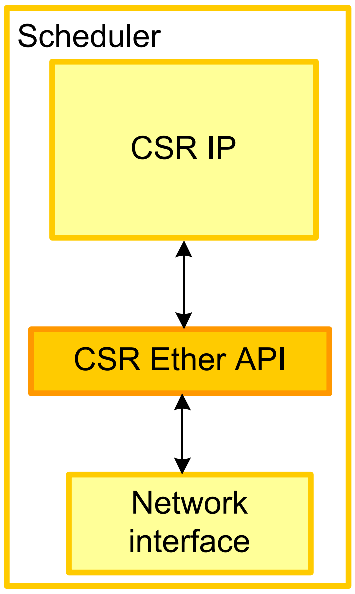


Figure 1: The CSR Ether API shown relative to the TCP/IP stack

3 Interface Description

The illustration below shows the lifecycle of an interface. The interface sends a CSR_IP_ETHER_IF_ADD_REQ to add the interface to the IP stack, which is confirmed by a CSR_IP_ETHER_IF_ADD_CFM that contains a result.

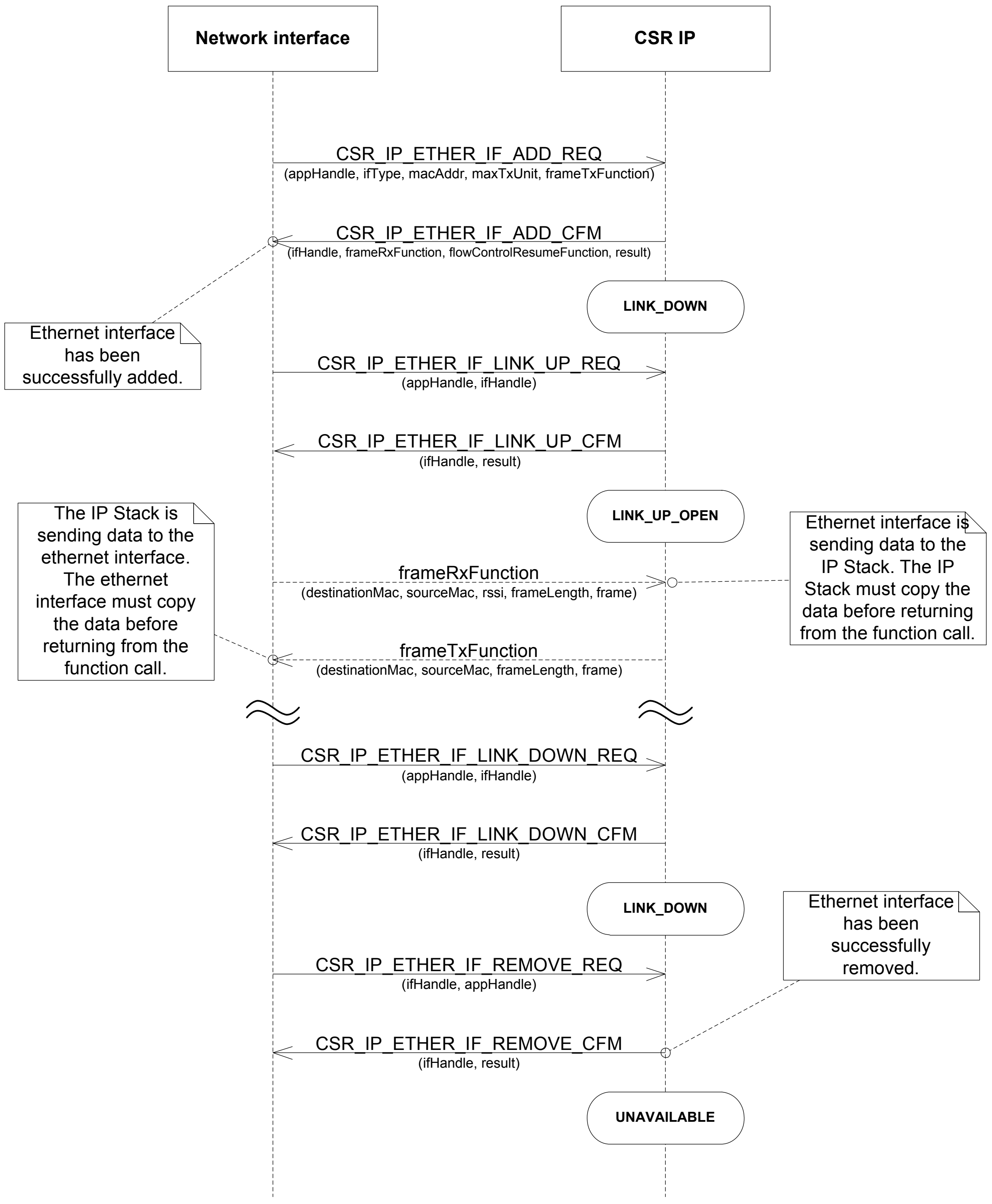

Network interface
CSR IP
CSR_IP_ETHER_IF_ADD_REQ
(appHandle, ifType, macAddr, maxTxUnit, frameTxFunction)
CSR_IP_ETHER_IF_ADD_CFM
(ifHandle, frameRxFunction, flowControlResumeFunction, result)
LINK_DOWN
Ethernet interface has been successfully added.
CSR_IP_ETHER_IF_LINK_UP_REQ
(appHandle, ifHandle)
CSR_IP_ETHER_IF_LINK_UP_CFM
(ifHandle, result)
LINK_UP_OPEN
The IP Stack is sending data to the ethernet interface. The ethernet interface must copy the data before returning from the function call.
Ethernet interface is sending data to the IP Stack. The IP Stack must copy the data before returning from the function call.
frameRxFunction
(destinationMac, sourceMac, rssi, frameLength, frame)
frameTxFunction
(destinationMac, sourceMac, frameLength, frame)
CSR_IP_ETHER_IF_LINK_DOWN_REQ
(appHandle, ifHandle)
CSR_IP_ETHER_IF_LINK_DOWN_CFM
(ifHandle, result)
LINK_DOWN
Ethernet interface has been successfully removed.
CSR_IP_ETHER_IF_REMOVE_REQ
(ifHandle, appHandle)
CSR_IP_ETHER_IF_REMOVE_CFM
(ifHandle, result)
UNAVAILABLE

Figure 2: A MSC of a CSR IP ETHER session

CSR_IP_ETHER_IF_LINK_UP_REQ is sent when the interface is ready to handle frames, and CSR_IP_ETHER_IF_LINK_UP_CFM is sent back when the link up request has been handled by the IP Stack. The IP Stack should refrain from transmitting frames, unless the link is up.

The CSR_IP_ETHER_IF_ADD_REQ contains a function pointer:

```
        CsrIpEtherFrameTxFunction frameTxFunction;
```

This function is called by the IP stack to transmit a frame.

Similarly, the CSR_IP_ETHER_IF_ADD_CFM contains a function pointer:

```
        CsrIpEtherFrameRxFunction frameRxFunction;
```

This pointer must be supplied by the IP stack. The frameRxFunction will be called to deliver a frame to the IP Stack.

The interface can be set in a link down state by the CSR_IP_ETHER_IF_LINK_DOWN_REQ. This typically happens when the connection to an access point is lost or a network cable is unplugged (only applicable in wired applications). The signal is confirmed by a CSR_IP_ETHER_IF_LINK_DOWN_CFM.

The interface can be removed completely from the IP stack by sending a CSR_IP_ETHER_REMOVE_REQ. This must be confirmed by sending a CSR_IP_ETHER_IF_REMOVE_CFM. On this event, the IP Stack must discard any information related to the interface. This includes the frameTxFunction pointer and the context associated with it.

4 Frame Encapsulation

It is possible to select the encapsulation format that is most suitable for the IP Stack. The encapsulation given in the CSR_IP_ETHER_IF_ADD_REQ is only a suggestion. The encapsulation format specified in the CSR_IP_ETHER_IF_ADD_CFM, which is sent by the IP Stack to the CSR Router, selects the encapsulation format to use for both reception and transmission of frames.

4.1 LLC/SNAP encapsulation

The default encapsulation is as specified in RFC 1042. This encapsulation is selected by setting the encapsulation to CSR_IP_ETHER_ENCAPSULATION_LLC_SNAP in the CSR_IP_ETHER_IF_ADD_CFM message.

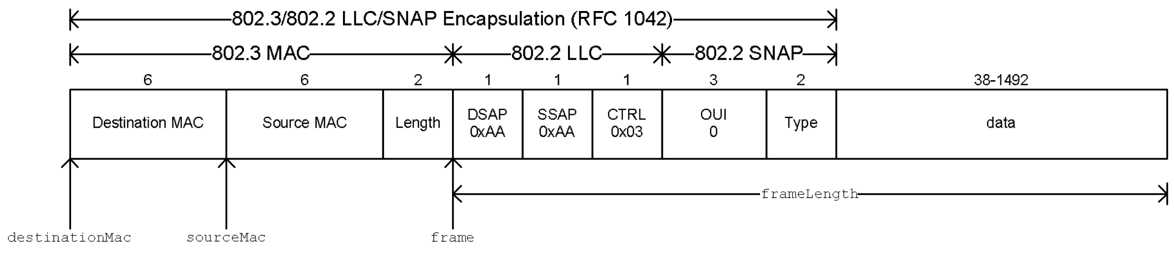


Figure 3: LLC/SNAP encapsulation

The figure shows how the arguments to the frameTxFunction and the frameRxFunction should be interpreted when using LLC/SNAP encapsulation.

4.2 Ethernet Encapsulation

Some IP stacks use the traditional ethernet frame format as specified by RFC 894. This encapsulation can be selected by setting the encapsulation to CSR_IP_ETHER_ENCAPSULATION_ETHERNET in the CSR_IP_ETHER_ADD_CFM message.

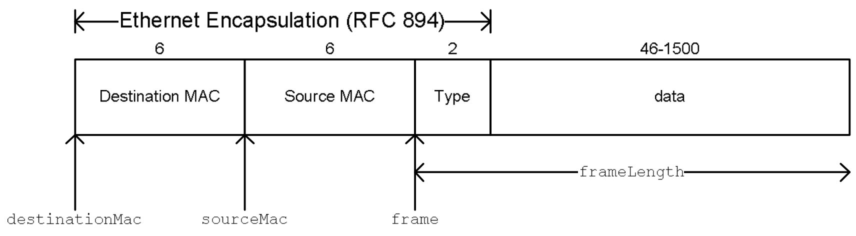


Figure 4: Ethernet encapsulation.

The figure shows how the arguments to the frameTxFunction and the frameRxFunction should be interpreted when using ethernet encapsulation.

5 Flow Control

In the basic form, the connection between the IP Stack and the ethernet interfaces can be in three states:

1. Before the interface is added and after an added interface is removed, the state is UNAVAILABLE.

2. When the interface is added the state is LINK_DOWN. This means that the interface exists, but no link is up, and no packages can be transmitted or received.

3. When the CSR_IP_ETHER_IF_LINK_UP_REQ has been received, the state is LINK_UP_OPEN.

The transitions between these three states are governed by the corresponding messages. The following state diagram shows the relationship.

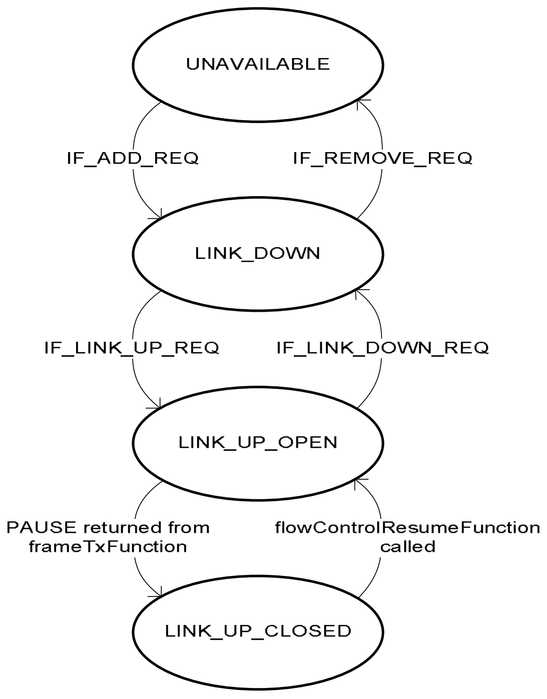


Figure 5: State diagram

Flow control adds a fourth state, denoted by LINK_UP_CLOSED. This state indicates that the link is up, but the flow control dictates that some frames cannot be transmitted due to congestion in the data path.

The transition to and from the LINK_UP_CLOSED state is governed by the priority argument of the IF_FLOW_CONTROL_PAUSE_REQ and IF_FLOW_CONTROL_RESUME_REQ messages. When the pause message is received by the IP stack, the priority argument will have at least one and possibly several of the bits specified by the CSR_IP_ETHER_PRIORITY_* defines set. This indicates that each of the corresponding priority levels are in flow control, and no frames of these priorities can be transmitted, until the relevant priority levels are resumed by one or several resume messages. Attempts to transmit frames of flow controlled priority will result in the frameTxFunction returning CSR_IP_ETHER_RESULT_NOT_SENT.

It is recommended to store the current flow control state in a local variable (of type CsrIpEtherPriority). When a pause message is received, the priority field of the request is bitwise or'ed into the local variable, causing the corresponding bits to be set:

```
flowControlState |= request->priority;
```

When the resume message is received, the priority field of the request is inverted and bitwise and'ed into the local variable, causing the corresponding bits to be cleared:

```
flowControlState &= ~request->priority;
```

Note that on reception of the pause message while in state LINK_UP_OPEN state, the state will change to LINK_UP_CLOSED. However, when receiving the resume message, the state is only transitioned to LINK_UP_OPEN again, if every bit in the local variable flowControlState has been cleared. Please also note, that when transitioning to the LINK_UP_OPEN state from the LINK_DOWN state by reception of a IF_LINK_UP_REQ message, no priority levels are in flow control by definition, and the local variable flowControlState must be cleared. I.e. it is not possible to transition directly from LINK_DOWN to LINK_UP_CLOSED, but it is possibly to transition directly from LINK_UP_CLOSED to LINK_DOWN by reception of a IF_LINK_DOWN_REQ.

Depending on the capabilities of the IP stack in question, the implementation may either perform fine-grained flow control on the individual priority levels, or simply consider the local variable flowControlState as a boolean, and simply refrain from transmitting all frames if any if the priority levels are in flow control. Any given implementation may also completely disregard the flow control mechanism and just attempt to transmit frames at any time. The frameTxFunction can return two different values:

1. CSR_RESULT_SUCCESS – The frame was successfully sent.

2. CSR_IP_ETHER_RESULT_NOT_SENT – The frame was not sent due to flow control.

Consequently, if frames are not sent, the IP stack must either queue the messages for transmission at a later time, or drop the frames.

CSR IP Ether Primitives

Primitives	Reference
CSR_IP_ETHER_IF_ADD_REQ/CFM	Section 5.1
CSR_IP_ETHER_IF_LINK_UP_REQ/CFM	Section 5.2
CSR_IP_ETHER_IF_LINK_DOWN_REQ/CFM	Section 5.3
CSR_IP_ETHER_IF_REMOVE_REQ/CFM	Section 5.4
CSR_IP_ETHER_IF_MULTICAST_ADDR_ADD_IND/RES	Section 5.5
CSR_IP_ETHER_IF_MULTICAST_ADDR_REMOVE_IND/RES	Section 5.6
CSR_IP_ETHER_IF_MULTICAST_ADDR_FLUSH_IND/RES	Section 5.7
CSR_IP_ETHER_IF_MULTICAST_ADDR_GET_IND/RES	Section 0

Table 1: List of CSR IP Ether Primitives

5.1 CSR_IP_ETHER_IF_ADD

Primitives \ Parameters	type	appHandle	ifHandle	ifType	mac	result	encapsulation	maxTxUnit	frameTxFunction	frameRxFunction	ifContext	ipContext
CSR_IP_ETHER_IF_ADD_REQ	✓	✓		✓	✓		✓	✓	✓		✓	
CSR_IP_ETHER_IF_ADD_CFM	✓		✓			✓	✓			✓		✓

Table 2: CSR_IP_ETHER_IF_ADD Primitives

Description

Add an interface to the IP stack, setting it into the LINK_DOWN state.

Parameters

type	CSR_IP_ETHER_IF_ADD_REQ/CFM
appHandle	The identity of the calling task.
ifHandle	A unique interface handle provided by the IP stack
ifType	CSR_IP_ETHER_IF_TYPE_WIRED or CSR_IP_ETHER_IF_TYPE_WIFI
mac	MAC address of the interface
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE or CSR_IP_ETHER_RESULT_NO_MORE_INTERFACES
encapsulation	The value passed in the request is the suggested encapsulation format for frames. The IP Stack will supply the actual encapsulation in the confirm, and this encapsulation format shall be used for both reception and transmission.
maxTxUnit	Maximum frame size that can be transmitted across a link. This value does not include headers.
frameTxFunction	Function pointer to function to be called by the IP Stack to transmit a frame.
frameRxFunction	Function pointer to function to be called by the ethernet interface push a received frame to the IP Stack.
ifContext	Opaque pointer to the context of the ethernet interface, passed in the CSR_IP_ETHER_IF_ADD_REQ to the IP Stack, which will pass it in every call to the corresponding CsrIpEtherFrameTxFunction.
ipContext	Opaque pointer to the context of the IP Stack, passed in the CSR_IP_ETHER_IF_ADD_CFM to the ethernet interface, which pass it in every call to the corresponding CsrIpEtherFrameRxFunction.

5.2 CSR_IP_ETHER_IF_LINK_UP

Primitives \ Parameters	type	appHandle	ifHandle	result
CSR_IP_ETHER_IF_LINK_UP_REQ	✓	✓	✓	
CSR_IP_ETHER_IF_LINK_UP_CFM	✓		✓	✓

Table 3: CSR_IP_ETHER_IF_LINK_UP Primitives

Description

Set the interface in LINK_UP_OPEN state. This indicates that frames can be transmitted by calling the frameTxFunction, and frames can be received by calls to the frameRxFunction.

Parameters

type	CSR_IP_ETHER_IF_LINK_UP_REQ/CFM
appHandle	The identity of the calling task.
ifHandle	A unique interface handle provided by the IP stack
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE or CSR_IP_ETHER_RESULT_INVALID_HANDLE

5.3 CSR_IP_ETHER_IF_LINK_DOWN

Primitives \ Parameters	type	appHandle	ifHandle	result
CSR_IP_ETHER_IF_LINK_DOWN_REQ	✓	✓	✓	
CSR_IP_ETHER_IF_LINK_DOWN_CFM	✓		✓	✓

Table 4: CSR_IP_ETHER_IF_LINK_DOWN Primitives

Description

Set the interface in LINK_DOWN state.

Parameters

type	CSR_IP_ETHER_IF_LINK_DOWN_REQ/CFM
appHandle	The identity of the calling task.
ifHandle	A unique interface handle provided by the IP stack
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE or CSR_IP_ETHER_RESULT_INVALID_HANDLE

5.4 CSR_IP_ETHER_IF_REMOVE

Primitives \ Parameters	type	appHandle	ifHandle	result
CSR_IP_ETHER_IF_REMOVE_REQ	✓	✓	✓	
CSR_IP_ETHER_IF_REMOVE_CFM	✓		✓	✓

Table 5: CSR_IP_ETHER_IF_REMOVE Primitives

Description

Remove an interface from the IP stack, setting the state to UNAVAILABLE.

Parameters

type	CSR_IP_ETHER_IF_REMOVE_REQ/CFM
appHandle	The identity of the calling task.
ifHandle	A unique interface handle provided by the IP stack
result	CSR_RESULT_SUCCESS, CSR_RESULT_FAILURE or CSR_IP_ETHER_RESULT_INVALID_HANDLE

5.5 CSR_IP_ETHER_IF_MULTICAST_ADDR_ADD

Primitives \ Parameters	type	ifHandle	result	multicastMacAddr
CSR_IP_ETHER_IF_MULTICAST_ADDR_ADD_IND	✓	✓		✓
CSR_IP_ETHER_IF_MULTICAST_ADDR_ADD_RES	✓	✓	✓	

Table 6: CSR_IP_ETHER_IF_MULTICAST_ADDR_ADD Primitives

Description

Used by the IP stack to inform the interface that a multicast address is subscribed to.

Parameters

type	CSR_IP_ETHER_IF_MULTICAST_ADDR_ADD_IND/RES
appHandle	The queue ID of the sending task
result	Result that shows whether the network interface could subscribe to the group
ifHandle	A unique interface handle provided by the IP stack
multicastMacAddr	The multicast address to subscribe to

5.6 CSR_IP_ETHER_IF_MULTICAST_ADDR_REMOVE

Primitives \ Parameters	type	ifHandle	result	multicastMacAddr
CSR_IP_ETHER_IF_MULTICAST_ADDR_REMOVE_IND	✓	✓		✓
CSR_IP_ETHER_IF_MULTICAST_ADDR_REMOVE_RES	✓	✓	✓	

Table 7: CSR_IP_ETHER_IF_MULTICAST_ADDR_REMOVE Primitives

Description

Used by the IP stack to inform the interface that a multicast address is no longer subscribed to.

Parameters

type	CSR_IP_ETHER_IF_MULTICAST_ADDR_REMOVE_IND/RES
appHandle	The queue ID of the sending task
result	Result that shows whether the network interface could unsubscribe from the group
ifHandle	A unique interface handle provided by the IP stack
multicastMacAddr	The multicast address to unsubscribe from

5.7 CSR_IP_ETHER_IF_MULTICAST_ADDR_FLUSH

Primitives \ Parameters	type	ifHandle	result
CSR_IP_ETHER_IF_MULTICAST_ADDR_FLUSH_IND	✓	✓	
CSR_IP_ETHER_IF_MULTICAST_ADDR_FLUSH_RES	✓	✓	✓

Table 8: CSR_IP_ETHER_IF_MULTICAST_ADDR_FLUSH Primitives

Description

Used by the IP stack to inform the interface that all multicast addresses are no longer subscribed to.

Parameters

type	CSR_IP_ETHER_IF_MULTICAST_ADDR_FLUSH_IND/RES
appHandle	The queue ID of the sending task
result	Result that shows whether the network interface could flush multicast subscriptions
ifHandle	A unique interface handle provided by the IP stack

5.8 CSR_IP_ETHER_IF_FLOW_CONTROL

Parameters / Primitives	type	ifHandle	priority
CSR_IP_ETHER_IF_FLOW_CONTROL_PAUSE_REQ	✓	✓	✓
CSR_IP_ETHER_IF_FLOW_CONTROL_RESUME_REQ	✓	✓	✓

Table 9: CSR_IP_ETHER_IF_FLOW_CONTROL Primitives

Description

Used by the ethernet interface to indicate pause and resume of flow control.

Parameters

type	CSR_IP_ETHER_IF_FLOW_CONTROL_PAUSE_REQ/RESUME_REQ
ifHandle	A unique interface handle provided by the IP stack
priority	A bit mask describing the priority levels that have been paused/resumed. Please note that this is not the current state of flow control, only bits corresponding to priority levels that have changed state (from pause to resume or vice versa) will be set. Use the recommended local variable method described in section 5 to retain information about the current flow control state.

6 CSR IP Ether Functions

6.1 CsrIpEtherFrameTxFunction

Description

This function type is passed as a function pointer in the CSR_IP_ETHER_IF_ADD_REQ from the ethernet interface to the IP Stack. The IP Stack must call this function to deliver a frame to the ethernet interface.

Parameters

destinationMac	Pointer to array of 6 byte containing the destination MAC address.
sourceMac	Pointer to array of 6 byte containing the source MAC address.
frameLength	Length (in byte) of the contents pointed at the frame pointer
frame	Pointer to the actual frame (not including MAC addresses). The specific contents depends on the encapsulation format chosen.
ifContext	This must be the ifContext member which was received together with the function pointer in the CSR_IP_ETHER_IF_ADD_REQ.

Returns

CSR_RESULT_SUCCESS	The frame was successfully transmitted.
CSR_IP_ETHER_RESULT_NOT_SENT	The frame was not sent. It may either be dropped or queued until flow control is removed trough a CSR_IP_ETHER_IF_FLOW_CONTROL_RESUME_REQ.

6.2 CsrIpEtherFrameRxFunction

Description

This function type is passed as a function pointer in the CSR_IP_ETHER_IF_ADD_CFM from the IP Stack to the ethernet interface. The ethernet interface will call this function to deliver a frame to the IP Stack.

Parameters

destinationMac	Pointer to array of 6 byte containing the destination MAC address.
sourceMac	Pointer to array of 6 byte containing the source MAC address.
rssi	Radio signal strength indication.
frameLength	Length (in byte) of the contents pointed at the frame pointer
frame	Pointer to the actual frame (not including MAC addresses). The specific contents depends on the encapsulation format chosen.
ipContext	This must be the ipContext which was received in the CSR_IP_ETHER_IF_ADD_CFM from the IP Stack.

7 Document References

Ref	Title
[SYN-FRW-IP-GU]	CSR Synergy Framework IP Archicture, gu-0002-ip-architecture

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
MSC	Message Sequence Chart

Document History

Revision	Date	History
1	22 FEB 09	Initial revision
2	30 NOV 09	Ready for release 2.0.0
3	20 APR 10	Ready for release 2.1.0
4	23 AUG 10	Documented multicast interface
5	DEC 10	Ready for release 3.0.0
6	Aug 11	Ready for release 3.1.0

TradeMarks, Patents and Licences

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.