DSIOETH Ethernet Interface

RTLib Reference

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
  Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/patches for software updates and patches.

Important Notice

Contents

About This Reference

Content

The DSIOETH Real-Time Library (RTLib) provides the C functions and macros you need to program the I/O Ethernet interface.

Supported Hardware

The following dSPACE systems are supported:

- DS1007 PPC Processor Board
- MicroLabBox

For more information, such as an overview of the supported network features and limitations, refer to General Information on the RTI Ethernet Blockset (RTI Ethernet Blockset Reference 🕮).

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
⚠ DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
⚠ WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
⚠ CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.
⍰	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
🕮	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

`%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>`

or

`%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>`

Documents folder A standard folder for user-specific documents.

`%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>`

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

`%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>`

Accessing dSPACE Help and PDF Files

After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the icon in dSPACE Help. The PDF opens on the first page.

Handling the Network Configuration

Purpose

To get information on handling the network configuration of the Ethernet I/O interface.

Where to go from here

Information in this section

IP Configuration Functions

Purpose

To get information on the functions used to specify the IP address configuration of an Ethernet I/O interface.

Where to go from here

Information in this section

DsIoEth_createAlias

Syntax

```
Int32 DsIoEth_createAlias(
      const char *strIp,
      const char *strNetMask,
      const char *strGateway)
```

Include file

`DsIoEth.h`

Purpose

To create an alternative IP configuration for the Ethernet interface.

Description

Use this function to assign more than one IP configuration consisting of IP address, netmask and gateway to the Ethernet interface.

The maximum number of IP aliases is limited to `DSIOETH_MAX_NUM_ALIASES`, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

Once aliases have been configured, they can not be changed until the application terminates.

Parameters

strIp Specifies the address of a string containing the IPv4 address of the alias. If NULL is specified, no alias will be created and the function returns an error.

strNetMask Specifies the address of a string containing the IPv4 netmask of the alias. If NULL is specified, no alias will be created and the function returns an error.

strGateway Specifies the address of a string containing the IPv4 gateway address. If NULL is specified, the alias will be created with the existing gateway configuration.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_setGatewayAddress 13
DsIoEth_setIpAddress 14
DsIoEth_setNetMask 14

DsIoEth_getGatewayAddress

Syntax

```
UInt32 DsIoEth_getGatewayAddress(void)
```

Include file

`DsIoEth.h`

Purpose

To get the gateway address of the Ethernet interface.

Parameters

None

Return value

This function returns the following values:

Returned Value	Meaning
An IPv4 address in network-byte order.	The IPv4 gateway address of the Ethernet interface.
`DSIOETH_INADDR_NONE`	The gateway address can not be read.

Related topics

References

DsIoEth_getIpAddress

Syntax

```
UInt32 DsIoEth_getIpAddress(void)
```

Include file

`DsIoEth.h`

Purpose

To get the IP address of the Ethernet interface.

Parameters

None

Return value

This function returns the following values:

Returned Value	Meaning
An IPv4 address in network-byte order.	The IP address of the Ethernet interface.
`DSIOETH_INADDR_NONE`	The IP address can not be read.

Related topics

References

DsIoEth_getNetMask

Syntax

```
UInt32 DsIoEth_getNetMask(void)
```

Include file

`DsIoEth.h`

Purpose

To get the netmask of the Ethernet interface.

Parameters

None

Return value

This function returns the following values:

Returned Value	Meaning
An IPv4 address in network-byte order.	The netmask of the Ethernet interface.
`DSIOETH_INADDR_NONE`	The netmask can not be read.

Related topics

References

DsIoEth_getGatewayAddress 11
DsIoEth_getIpAddress 12
DsIoEth_ntohl 71
DsIoEth_setNetMask 14

DsIoEth_setGatewayAddress

Syntax

```
void DsIoEth_setGatewayAddress(const char *strGateway)
```

Include file

`DsIoEth.h`

Purpose

To set the gateway address of the Ethernet interface.

Parameters

strGateway Specifies the address of a string containing the IPv4 gateway address. If NULL is specified, the configuration will not be changed.

Return value None

Related topics

References

DsIoEth_setIpAddress

Syntax

```
void DsIoEth_setIpAddress(const char *strIp)
```

Include file `DsIoEth.h`

Purpose To set the IP address of the Ethernet interface.

Parameters **strIp** Specifies the address of a string containing the IPv4 address. If NULL is specified, the configuration will not be changed.

Return value None

Related topics

References

DsIoEth_setNetMask

Syntax

```
void DsIoEth_setNetMask(const char *strNetMask)
```

Include file `DsIoEth.h`

Purpose	To set the netmask of the Ethernet interface.
Parameters	**strNetMask** Specifies the address of a string containing the IPv4 netmask. If NULL is specified, the configuration will not be changed.
Return value	None
Related topics	References

DHCP Functions

Purpose

To get information on the functions used for dynamic IP configuration via DHCP.

Where to go from here

Information in this section

DsIoEth_getDhcpClientState

Syntax

```
Int32 DsIoEth_getDhcpClientState(void)
```

Include file

`DsIoEth.h`

Purpose

To get the state of the local DHCP client.

Description

Use this function to determine the state of the local DHCP client as an element of the `DsEIoEthDhcpClientState` enumeration, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

Parameters

None

Return value

This function returns the following DHCP client states:

DHCP Client State	Meaning
DSIOETH_DHCP_CLIENT_STATE_OFF	The DHCP client has not yet been started.
DSIOETH_DHCP_CLIENT_STATE_RUNNING	The DHCP client has been started and is ready to receive the IP address from the DHCP server.
DSIOETH_DHCP_CLIENT_STATE_BOUND	The DHCP client has received its IP address from the DHCP server. The network interface is now ready to send and receive data with its new IP address configuration.

DHCP Client State	Meaning
DSIOETH_DHCP_CLIENT_STATE_TIMEOUT	DHCP client did not receive an IP address within the timeout specified by **`DsIoEth_startDhcpClient`**.

Related topics

References

DsIoEth_startDhcpClient..... 17

DsIoEth_startDhcpClient

Syntax

```
void DsIoEth_startDhcpClient(UInt32 Timeout)
```

Include file

`DsIoEth.h`

Purpose

To start the local DHCP client to obtain an IP address configuration.

Description

In networks with dynamic host configuration, use this function to start the DHCP client to get an IP address configuration from a DHCP server.

You can use **`DsIoEth_getDhcpClientState`** to monitor this.

Parameters

Timeout Specifies the DHCP client timeout in milliseconds. As a default, you can use `20000`, which is also defined as `DSIOETH_DHCP_CLIENT_DEFAULT_TIMEOUT_MSEC` in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

Return value

None

Related topics

References

DsIoEth_getDhcpClientState..... 16

Communication State Functions

Purpose

To get information on the functions used to get the communication state of an Ethernet I/O interface.

Where to go from here

Information in this section

DsIoEth_getLinkState

Syntax

```
Int32 DsIoEth_getLinkState(UInt32 InterfaceNumber)
```

Include file

`DsIoEth.h`

Purpose

To get the link state of the Ethernet interface.

Description

Use this function to determine the link state of a specified Ethernet interface as an element of the `DsEIoEthLinkState` enumeration, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`. The link state tells you whether the interface is physically connected and if so, to which speed and duplex mode.

Parameters

InterfaceNumber Specifies the interface whose link state has to be determined. `DSIOETH_DEFAULT_INTERFACE` is the default value for the first Ethernet connector.

Return value

This function returns the following link states:

Link State	Meaning
DSIOETH_LINK_STATE_NOT_CONNECTED	The specified interface is not connected.
DSIOETH_LINK_STATE_10BASE_T_HALF_DUPLEX	The specified interface is linked to a 10BASE-T[1)] connection in half duplex mode.
DSIOETH_LINK_STATE_10BASE_T_FULL_DUPLEX	The specified interface is linked to a 10BASE-T[1)] connection in full duplex mode.
DSIOETH_LINK_STATE_100BASE_TX_HALF_DUPLEX	The specified interface is linked to a 100BASE-TX[1)] connection in half duplex mode.
DSIOETH_LINK_STATE_100BASE_T4	The specified interface is linked to a 100BASE-T4[1)] connection.
DSIOETH_LINK_STATE_100BASE_TX_FULL_DUPLEX	The specified interface is linked to a 100BASE-TX[1)] connection in full duplex mode.
DSIOETH_LINK_STATE_1000BASE_T_HALF_DUPLEX	The specified interface is linked to a 1000BASE-T[1)] connection in half duplex mode.
DSIOETH_LINK_STATE_1000BASE_T_FULL_DUPLEX	The specified interface is linked to a 1000BASE-T[1)] connection in full duplex mode.
DSIOETH_LINK_STATE_UNKNOWN	An error occurred and the link state can not be detected.

[1)] Adapted and reprinted with permission from IEEE. Copyright IEEE 2018. All rights reserved.

Related topics

References

DsIoEth_getServiceAliveState..... 19

DsIoEth_getServiceAliveState

Syntax

```
Int32 DsIoEth_getServiceAliveState(void)
```

Include file

`DsIoEth.h`

Purpose

To get the state of the Ethernet I/O service.

Description

Use this function to determine the state of the Ethernet I/O service as an element of the `DsEIoEthServiceAliveState` enumeration, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`. This is, for example, useful for debugging purposes to detect an unexpected termination of the Ethernet I/O service.

Parameters

None

Return value

This function returns the following service states:

Service State	Meaning
DSIOETH_SERVICE_ALIVE_STATE_RUNNING	The service is running.
DSIOETH_SERVICE_ALIVE_STATE_OFF	The service is not running.

Related topics

References

DsIoEth_getLinkState..... 18

Handling Ethernet Socket Connections

Purpose

To get information on using Ethernet socket connections to transfer data.

Where to go from here

Information in this section

Socket Configuration Functions

Purpose

To get information on the functions to configure Ethernet sockets.

Where to go from here

Information in this section

DsIoEth_create

Syntax

```
Int32 DsIoEth_create(
      UInt32 ConnectionId,
      const DsSSockAddr *pAddr,
      UInt32 AddrLen,
      DsEIoEthProtocol Protocol,
      DsEIoEthFlag Flags)
```

Include file

`DsIoEth.h`

Purpose

To create an Ethernet socket with a specified connection identifier.

Description

Use this function to create an Ethernet socket for a specified connection identifier and parameterize it with a socket address and a protocol.

A socket address consists of the local IP address and a port number.

Via the protocol parameter, you can specify to create a TCP or a UDP socket and to use it on the client or on the server side.

You can then open a connection via a created socket using **`DsIoEth_open`**.

Parameters

ConnectionId Specifies the unique connection identifier to create a socket for in the range of `0 … (DSIOETH_MAX_CONNECTION_IDS-1)`, i.e., `0 … 259`.

pAddr Specifies the address of a structure containing the local socket address to be used.

AddrLen Specifies the length of the socket address in bytes.

Protocol Specifies the protocol (UDP/TCP) and the role (client/server) to use.
The following symbols are defined:

Predefined Symbol	Meaning
DSIOETH_PROTO_UDP_CLIENT	Creates a UDP client socket.
DSIOETH_PROTO_UDP_SERVER	Creates a UDP server socket.
DSIOETH_PROTO_TCP_CLIENT	Creates a TCP client socket.
DSIOETH_PROTO_TCP_SERVER	Creates a TCP server socket.

Flags Specifies flags to configure the connection behavior.
The following flags are defined:

Flag	Meaning
DSIOETH_FLAG_NONE	No flag is set.
DSIOETH_FLAG_TCP_NODELAY	Disables the Nagle algorithm, which is enabled by default. With this flag set, the TCP/IP stack tries to send outgoing data as soon as possible.
DSIOETH_FLAG_TCP_NO_PENDING_CONN	Disables accepting connections if a socket connection was closed using **`DsIoEth_close`**. This flag is only relevant for TCP server socket connections.
DSIOETH_FLAG_IRQ_ACK_MODE	If this flag is set, any interrupt blocks the generation of further interrupts of the same extended interrupt identifier until the interrupt is acknowledged. The acknowledgement must be done by using **`DsIoEth_acknowledgeInt`** at the end of the interrupt service routine.

Return value

This function returns the following values:

Returned Value	Meaning
The specified connection identifier.	The socket was created.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_createFd

Syntax

```
Int32 DsIoEth_createFd(
      const DsSSockAddr *pAddr,
      UInt32 AddrLen,
      DsEIoEthProtocol Protocol,
      DsEIoEthFlag Flags)
```

Include file

`DsIoEth.h`

Purpose

To create an Ethernet socket with a dynamically generated file descriptor.

Description

Use this function to create an Ethernet socket with the specified socket address and protocol. You can use the returned file descriptor in subsequent call as the connection identifier.

A socket address consists of the local IP address and a port number.

Via the protocol parameter, you can specify to create a TCP or a UDP socket and to use it on the client or on the server side.

You can open a connection via a socket using **`DsIoEth_open`**.

Parameters

pAddr Specifies the address of a structure containing the local socket address to be used.

AddrLen Specifies the length of the socket address in bytes.

Protocol Specifies the protocol (UDP/TCP) and the role (client/server) to use. The following symbols are defined:

Predefined Symbol	Meaning
DSIOETH_PROTO_UDP_CLIENT	Creates a UDP client socket.
DSIOETH_PROTO_UDP_SERVER	Creates a UDP server socket.

Predefined Symbol	Meaning
DSIOETH_PROTO_TCP_CLIENT	Creates a TCP client socket.
DSIOETH_PROTO_TCP_SERVER	Creates a TCP server socket.

Flags Specifies flags to configure the connection behavior.

The following flags are defined:

Flag	Meaning
DSIOETH_FLAG_NONE	No flag is set.
DSIOETH_FLAG_TCP_NODELAY	Disables the Nagle algorithm, which is enabled by default. With this flag set, the TCP/IP stack tries to send outgoing data as soon as possible.
DSIOETH_FLAG_TCP_NO_PENDING_CONN	Disables accepting connections if a socket connection was closed using **`DsIoEth_close`**. This flag is only relevant for TCP server socket connections.
DSIOETH_FLAG_IRQ_ACK_MODE	If this flag is set, any interrupt blocks the generation of further interrupts of the same extended interrupt identifier until the interrupt is acknowledged. The acknowledgement must be done by using **`DsIoEth_acknowledgeInt`** at the end of the interrupt service routine.

Return value This function returns the following values:

Returned Value	Meaning
The specified file descriptor.	The socket was created.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_create 22
DsIoEth_destroy 25
DsIoEth_open 34

DsIoEth_destroy

Syntax

```
Int32 DsIoEth_destroy(UInt32 ConnectionId)
```

Include file `DsIoEth.h`

Purpose

To destroy an Ethernet socket and free the related resources.

Description

Use this function to close a socket connection that you created with **`DsIoEth_create`** on page 22. All resources related to the socket connection are freed.

After the execution of this function, the connection identifier can be used again to create a new socket with **`DsIoEth_create`**.

Note

Do not use DsIoEth_destroy() with file descriptors that are returned by **`DsIoEth_createFd`** on page 24.

Parameters

ConnectionId Specifies the unique connection identifier of the socket to be destroyed.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_create..... 22
DsIoEth_createFd..... 24

DsIoEth_getMaxNumberOfConnections

Syntax

```
UInt32 DsIoEth_getMaxNumberOfConnections(void)
```

Include file

`DsIoEth.h`

Purpose

To get the maximum number of connections that can be created.

Parameters

None

Return value

This function returns the maximum number of connections that can be created.

Related topics

References

DsIoEth_create.....22

DsIoEth_queryArpEntry

Syntax

```
Int32 DsIoEth_queryArpEntry(
            UInt32 ConnectionId,
            DsSSockAddr *pAddr,
            UInt32 AddrLen)
```

Include file

`DsIoEth.h`

Purpose

To resolve the IP address of a remote system.

Description

Use this function to determine the physical address of a remote system and make it known to the local system for a faster subsequent access.

The specified connection is used to broadcast an ARP request for the remote system that is specified by its IP address. If the remote system answered with an ARP reply, its physical address is added to the internal address resolution table of the local system.

Tip

This procedure is implicitly executed for any access to an unknown system. By calling DsIoEth_queryArpEntry() you can separate address resolution to an initialization phase.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pAddr Specifies the address of a structure containing the remote socket address to be used.

AddrLen Specifies the length of the socket address in bytes.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_open.....34

DsIoEth_setAddrFilter

Syntax

```
Int32 DsIoEth_setAddrFilter(
      UInt32 ConnectionId,
      DsSSockAddr *pAddr,
      UInt32 AddrLen)
```

Include file

`DsIoEth.h`

Purpose

To configure an address filter for a socket.

Description

Use this function to configure the remote socket address for a local socket to accept data from. The local socket is specified by its connection identifier.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pAddr Specifies the address of a structure containing the remote socket address to be used.

If the port number of this socket address is 0, incoming packets or connection requests from any remote port are accepted. Otherwise, only packets or connection requests with the specified remote port are accepted.

If the IP address of this socket address is 0, incoming packets or connection requests from any remote IP address are accepted. Otherwise, only packets or connection requests with the specified remote IP address are accepted.

AddrLen Specifies the length of the socket address in bytes.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_setRecvFrameSize 29

DsIoEth_setRecvFrameSize

Syntax

```
Int32 DsIoEth_setRecvFrameSize(
      UInt32 ConnectionId,
      UInt32 FrameSize,
      DsEIoEthRecvMode Mode)
```

Include file

`DsIoEth.h`

Purpose

To configure the frame size used by the application.

Description

Use this function to configure the following Ethernet parameters used by your application:

- The frame size for TCP and UDP.
- The receive mode for UDP. This parameter configures the behavior for packets smaller than the frame size.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

FrameSize Specifies the length of a receive frame in bytes.

Mode Specifies the receive mode for UDP sockets. This parameter is ignored by TCP sockets.

The following modes are defined:

Mode	Meaning
DSIOETH_RECV_MODE_EQUAL_SIZE	Only UDP packets equal to the specified frame size will be received and signaled.
DSIOETH_RECV_MODE_MAXIMUM_SIZE	Only UDP packets less than or equal to the specified frame size will be received and signaled.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_setAddrFilter..... 28

Socket Connection Handling Functions

Purpose

To get information on the functions used to establish and to terminate Ethernet connections.

Where to go from here

Information in this section

DsIoEth_accept

Syntax

```
Int32 DsIoEth_accept(
      UInt32 ConnectionId,
      DsSSockAddr *pAddr,
      UInt32 *pAddrLen)
```

Include file

`DsIoEth.h`

Purpose

To wait for a TCP client to connect.

Description

Use this function to let a local socket accept incoming connection requests from a remote client.

The local socket must have been created as a server socket using **`DsIoEth_create`** or **`DsIoEth_createFd`**. After that, it must be opened using **`DsIoEth_open`**.

You must call this function multiple times until a client has established a connection to the specified server socket.

Parameters

ConnectionId Defines the unique connection identifier of the local server socket.

pAddr Specifies the address of a structure that will contain the client socket address after connection has been established. You must allocate this structure before you call this function. The returned socket will be truncated if this structure is too small.

pAddrLen Specifies the address of a variable containing the length of the socket address structure at `pAddr`.

Before calling this function, you must initialize this variable with the current length. If the length has increased after the call, this indicates that the client is incompatible with this server socket.

Return value

This function returns the following values:

Returned Value	Meaning
`-1`	No client has been connected yet.
A positive integer value.	A connection has been established.

Related topics

References

DsIoEth_close

Syntax

```
Int32 DsIoEth_close(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To close a socket.

Description

Use this function to close a socket without freeing its resources. This makes it possible to reopen the socket again using **`DsIoEth_open`** without calling **`DsIoEth_create`** beforehand.

If you want to free the socket's resources, you can call **`DsIoEth_destroy`**.

Parameters

ConnectionId Specifies the unique connection identifier of the socket to be closed.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_create 22
DsIoEth_createFd 24
DsIoEth_destroy 25
DsIoEth_open 34

DsIoEth_connect

Syntax

```
Int32 DsIoEth_connect(
      UInt32 ConnectionId,
      DsSSockAddr *pAddr,
      UInt32 AddrLen)
```

Include file

`DsIoEth.h`

Purpose

To connect to a TCP server.

Description

Use this function to let a local socket send a connection request to a remote server.

The local socket must have been created as a client socket using **`DsIoEth_create`** or **`DsIoEth_createFd`**. After that, it must have been opened using **`DsIoEth_open`**.

After calling this function once, you must check the connection state before data can be sent or received. There are two ways to do this:

- You can use **`DsIoEth_getConnectionState`** to verify, that the connection is established.
- You can poll incoming events using **`DsIoEth_getMgmtEvent`**.

Parameters

ConnectionId Specifies the unique connection identifier of the local client socket to use.

pAddr Specifies the address of a structure containing the server socket address you want to connect to.

AddrLen Specifies the size of the socket address in `pAddr` in bytes.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_open

Syntax

```
Int32 DsIoEth_open(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To open a socket.

Description

Use this function to open a socket that you created using **`DsIoEth_create`** or **`DsIoEth_createFd`**.

Parameters

ConnectionId Specifies the unique connection identifier of the socket to open.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

Socket State Functions

Purpose

To get information on the functions used to get the state of Ethernet sockets and connections.

Where to go from here

Information in this section

DsIoEth_getConnectionState

Syntax

```
Int32 DsIoEth_getConnectionState(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To get the connection state of a socket.

Description

Use this function to determine a socket connection state, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`. The socket is specified by its connection identifier.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

This function returns one of the following socket connection states:

Socket Connection State	Meaning
DSIOETH_CONNECTION_NOT_ESTABLISHED	The connection is not established.
DSIOETH_CONNECTION_ESTABLISHED	The connection is established.

Related topics

References

DsIoEth_getLinkState..... 18
DsIoEth_getPortState..... 37
DsIoEth_getSocketState..... 39

DsIoEth_getPortState

Syntax

```
Int32 DsIoEth_getPortState(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To get the port state of a socket.

Description

Use this function to determine a socket's port state, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`. The socket is specified by its connection identifier.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

This function returns the following port states:

Port State	Meaning
DSIOETH_PORT_CLOSED	The port is in closed state.
DSIOETH_PORT_OPEN	The port is in open state.

Related topics

References

DsIoEth_getConnectionState.....36
DsIoEth_getLinkState.....18
DsIoEth_getSocketState.....39

DsIoEth_getRecvFramesDropped

Syntax

```
Int32 DsIoEth_getRecvFramesDropped(
      UInt32 ConnectionId,
      UInt32 *pDroppedByFilter,
      UInt32 *pDroppedByOverflow)
```

Include file

`DsIoEth.h`

Purpose

To get the number of dropped received frames.

Description

Use this function to determine how many received frames were dropped due to the RecvFrameSize filter and how many due to an overflow of the receive FIFO buffer.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pDroppedByFilter Specifies the address of the variable to store the returned number of frames dropped due to the RecvFrameSize filter.

If NULL is specified, this value is ignored.

pDroppedByOverflow Specifies the address of the variable to store the returned number of frames dropped due to FIFO overflow.

If NULL is specified, this value is ignored.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_setAddrFilter..... 28

DsIoEth_getSocketState

Syntax

```
Int32 DsIoEth_getSocketState(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To get the state of a socket.

Description

Use this function to determine the state of a socket as an element of the `DsEIoEth_SocketState` enumeration, which is defined in `DsIoEth_SocketState.h`.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

This function returns the following socket states:

Socket State	Meaning
SOCKET_STATE_UNKNOWN	This is the initialization value.
SOCKET_STATE_IDLE	The specified socket has been created.
SOCKET_STATE_OPEN	The specified socket has been opened.
SOCKET_STATE_ACCEPT	The specified server socket has been opened and is waiting for an incoming connection.
SOCKET_STATE_CONNECTED	The specified client or server socket has an established connection.
SOCKET_STATE_CLOSED	The specified socket is closed.
SOCKET_STATE_PENDING	The specified server socket is not open yet, i.e., it is not in the SOCKET_STATE_ACCEPT state, but a connection request is already pending.
SOCKET_STATE_CONNECT	The specified client socket is not connected yet, i.e., it is not in the SOCKET_STATE_CONNECTED state, and a connection request is still pending.

Related topics

References

Data Transfer Functions

Purpose

To get information on the functions used to read from or write to a socket connection.

Where to go from here

Information in this section

DsIoEth_recv

Syntax

```
Int32 DsIoEth_recv(
      UInt32 ConnectionId,
      void *pBuf,
      Int32 Len,
      Int32 Flags)
```

Include file

`DsIoEth.h`

Purpose

To receive data from a socket connection.

Description

Use this function to receive data:

- From the client or the server side of a TCP socket connection.
- From the client side of a UDP socket connection, if **`DsIoEth_connect`** on page 33 has been executed for this socket.

The received data is written to a buffer that you must allocate before calling this function.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pBuf Specifies the address of the buffer to store the received data.

If a packet is too long to fit in the supplied receive buffer:

- On UDP socket connections, excess bytes will always be discarded.
- On TCP socket connections, excess bytes will remain in the internal receive buffer and can be read with further calls of this function.

Len Specifies the length of the buffer in bytes.

Flags Specifies flags to configure the behavior of the data transfer.

The following flags are defined:

Flag	Meaning
DSIOETH_FLAG_NONE	No flag is set.
DSIOETH_FLAG_TCP_COMPLETE_MSG	▪ If the internal receive buffer contains less than Len bytes and this flag is set, no data is copied to the user specified buffer. ▪ If the internal receive buffer contains less than Len bytes and this flag is not set, all available data is copied o the user specified buffer.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The connection was closed or was invalid.
`-1`	No data was transfered.
A positive integer value.	The number of bytes that were transferred.

Related topics

References

DsIoEth_recvDataAvail....42
DsIoEth_recvfrom....43
DsIoEth_send....45
DsIoEth_sendto....46

DsIoEth_recvDataAvail

Syntax

```
Int32 DsIoEth_recvDataAvail(UInt32 ConnectionId)
```

Include file `DsIoEth.h`

Purpose To get the expected length of the data to be received.

Description Use this function to get the minimum number of bytes that can be expected when calling **`DsIoEth_recv`**.

Parameters **ConnectionId** Specifies the unique connection identifier to be used.

Return value This function returns the following values:

Returned Value	Meaning
`-1`	The function terminated with an error.
A positive integer value.	The number of bytes to be received.

Related topics

References

DsIoEth_recv.....41

DsIoEth_recvfrom

Syntax

```
Int32 DsIoEth_recvfrom(
      UInt32 ConnectionId,
      void *pBuf,
      Int32 Len,
      Int32 Flags,
      DsSSockAddr *pSrcAddr,
      UInt32 *pAddrLen)
```

Include file `DsIoEth.h`

Purpose To receive data and the sender socket address from a socket connection.

Description

Use this function to receive data from a UDP or TCP socket and write it to the specified buffer that you allocated beforehand. The socket address of the sender is also returned.

For TCP sockets the source socket address is ignored.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pBuf Specifies the address of the buffer to store the received data.

If a packet is too long to fit in the supplied receive buffer:

- On UDP socket connections, excess bytes will always be discarded.
- On TCP socket connections, excess bytes will remain in the internal receive buffer and can be read with further calls of this function.

Len Specifies the length of the buffer in bytes.

Flags Specifies flags to configure the behavior of the data transfer.

The following flags are defined:

Flag	Meaning
DSIOETH_FLAG_NONE	No flag is set.
DSIOETH_FLAG_TCP_COMPLETE_MSG	▪ If the internal receive buffer contains less than Len bytes and this flag is set, no data is copied to the user specified buffer. ▪ If the internal receive buffer contains less than Len bytes and this flag is not set, all available data is copied o the user specified buffer.

pSrcAddr Specifies the address of the structure to contain the remote socket address from which the data is received. You must allocate this structure before you call of this function. The returned socket address will be truncated if this buffer is too small.

pAddrLen Specifies the address of a variable containing the length of the socket address structure at `pSrcAddr`.

Before calling this function, you must initialize this variable with the current length. If the length has increased after the call, this indicates that the remote socket is incompatible with the local socket.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The connection was closed or was invalid.
`-1`	No data was transfered.
A positive integer value.	The number of bytes that were transferred.

Related topics

References

DsIoEth_send

Syntax

```
Int32 DsIoEth_send(
      UInt32 ConnectionId,
      const void *pBuf,
      Int32 Len,
      Int32 Flags)
```

Include file

`DsIoEth.h`

Purpose

To send data via a socket connection.

Description

Use this function to read data from the specified buffer and send it to a remote socket:

- Via the client or the server side of a TCP socket connection.
- Via the client side of a UDP socket connection, if **`DsIoEth_connect`** on page 33 has been executed for this socket.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pBuf Specifies the address of the buffer containing the data to be sent.

Len Specifies the length of the buffer in bytes.

Flags Specifies flags to configure the behavior of the data transfer.
The following flags are defined:

Flag	Meaning
DSIOETH_FLAG_NONE	No flag is set.
DSIOETH_FLAG_TCP_COMPLETE_MSG	▪ If the length of the internal transmit buffer is too small and this flag is set, no data is transferred. ▪ If the length of the internal transmit buffer is too small and this flag is not set, the transferred data is truncated.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The connection was closed or was invalid.
`-1`	No data was transfered.
A positive integer value.	The number of bytes that were transferred.

Related topics

References

DsIoEth_recv.....41
DsIoEth_recvfrom.....43
DsIoEth_sendto.....46

DsIoEth_sendto

Syntax

```
Int32 DsIoEth_sendto(
      UInt32 ConnectionId,
      const void *pBuf,
      Int32 Len,
      Int32 Flags,
      const DsSSockAddr *pDestAddr,
      UInt32 AddrLen)
```

Include file

`DsIoEth.h`

Purpose

To send data to a specified socket address via a UDP or TCP socket connection.

Description

Use this function to read data from the specified buffer and send it via a socket connection to the specified destination socket address.

For TCP sockets the destination socket address is ignored.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

pBuf Specifies the address of the buffer containing the data to be sent.

Len Specifies the length of the buffer in bytes.

Flags Specifies flags to configure the behavior of the data transfer.

The following flags are defined:

Flag	Meaning
DSIOETH_FLAG_NONE	No flag is set.
DSIOETH_FLAG_TCP_COMPLETE_MSG	▪ If the length of the internal transmit buffer is too small and this flag is set, no data is transferred. ▪ If the length of the internal transmit buffer is too small and this flag is not set, the transferred data is truncated.

pDestAddr Specifies the address of a structure that contains the remote socket address to which the data is to be sent. For TCP sockets, this parameter is ignored.

AddrLen Specifies the length of the socket address in bytes. For TCP sockets, this parameter is ignored.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The connection was closed or was invalid.
`-1`	No data was transfered.
A positive integer value.	The number of bytes that were transferred.

Related topics

References

DsIoEth_recv....41
DsIoEth_recvfrom....43
DsIoEth_send....45

Handling Interrupts

Purpose

To get information on controlling data transfer using interrupts.

Where to go from here

Information in this section

Common Interrupt Control Functions

Purpose

To get information on the use of common interrupt control functions.

Where to go from here

Information in this section

DsIoEth_acknowledgeInt

Syntax

```
void DsIoEth_acknowledgeInt(UInt32 ExtendedIntId)
```

Include file

`DsIoEth.h`

Purpose

To acknowledge an interrupt.

Description

Use this function to acknowledge the specified interrupt.

This is relevant only, if the socket of the connection has been created with the `DSIOETH_FLAG_IRQ_ACK_MODE` flag set.

Parameters

ExtendedIntId Specifies the extended interrupt identifier to be used. This is commonly done via a symbolic name that is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

An extended interrupt identifier specifies the following:

- Whether it is a receive (RX), a send (TX) or a management (MGMT) interrupt
- The connection identifier in the range `0` … `255` for which the interrupt occurs

The following extended interrupt identifiers are defined:

Extended Interrupt Identifier	Meaning
DSIOETH_RX_INT_0	RxData interrupt for connection identifier 0.
…	…
DSIOETH_RX_INT_255	RxData interrupt for connection identifier 255.
DSIOETH_TX_INT_0	TxSent interrupt for connection identifier 0.
…	…
DSIOETH_TX_INT_255	TxSent interrupt for connection identifier 255.
DSIOETH_MGMT_INT_0	Management interrupt for connection identifier 0.
…	…
DSIOETH_MGMT_INT_255	Management interrupt for connection identifier 255.

Return value

None

Related topics

References

DsIoEth_disableInt

Syntax

```
void DsIoEth_disableInt(UInt32 ExtendedIntId)
```

Include file

`DsIoEth.h`

Purpose

To disable an interrupt.

Description

Use this function to disable the specified interrupt.

After this call is executed, there are no more calls for the interrupt handler routine of the specified extended interrupt identifier. You can use **`DsIoEth_enableInt`** to re-enable the handling of this interrupt.

Parameters

ExtendedIntId Specifies the extended interrupt identifier to be used. This is commonly done via a symbolic name that is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

An extended interrupt identifier specifies the following:

- Whether it is a receive (RX), a send (TX) or a management (MGMT) interrupt
- The connection identifier in the range `0 … 255` for which the interrupt occurs

The following extended interrupt identifiers are defined:

Extended Interrupt Identifier	Meaning
DSIOETH_RX_INT_0	RxData interrupt for connection identifier 0.
…	…
DSIOETH_RX_INT_255	RxData interrupt for connection identifier 255.
DSIOETH_TX_INT_0	TxSent interrupt for connection identifier 0.
…	…
DSIOETH_TX_INT_255	TxSent interrupt for connection identifier 255.
DSIOETH_MGMT_INT_0	Management interrupt for connection identifier 0.
…	…
DSIOETH_MGMT_INT_255	Management interrupt for connection identifier 255.

Return value

None

Related topics

References

DsIoEth_enableInt

Syntax

```
void DsIoEth_enableInt(UInt32 ExtendedIntId)
```

Include file

`DsIoEth.h`

Purpose

To enable an interrupt.

Description

Use this function to enable the specified interrupt.

After the execution of this call, the interrupt handler routine for the specified extended interrupt identifier is executed if the interrupt occurs.

Parameters

ExtendedIntId Specifies the extended interrupt identifier to be used. This is commonly done via a symbolic name that is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

An extended interrupt identifier specifies the following:

- Whether it is a receive (RX), a send (TX) or a management (MGMT) interrupt
- The connection identifier in the range `0 … 255` for which the interrupt occurs

The following extended interrupt identifiers are defined:

Extended Interrupt Identifier	Meaning
DSIOETH_RX_INT_0	RxData interrupt for connection identifier 0.
…	…
DSIOETH_RX_INT_255	RxData interrupt for connection identifier 255.
DSIOETH_TX_INT_0	TxSent interrupt for connection identifier 0.
…	…
DSIOETH_TX_INT_255	TxSent interrupt for connection identifier 255.
DSIOETH_MGMT_INT_0	Management interrupt for connection identifier 0.
…	…
DSIOETH_MGMT_INT_255	Management interrupt for connection identifier 255.

Return value

None

Related topics

References

DsIoEth_acknowledgeInt 50
DsIoEth_disableInt 51
DsIoEth_installIntHandler 53

DsIoEth_installIntHandler

Syntax

```
DsTIoEth_IntHandler DsIoEth_installIntHandler(
                    UInt32 ExtendedIntId,
                    DsTIoEth_IntHandler IntHandler)
```

Include file

`DsIoEth.h`

Purpose

To install an interrupt handler routine to be called at a specific interrupt.

Description

Use this function to configure the interrupt handler routine i.e., the function, to call if a specific interrupt occurs.

Parameters

ExtendedIntId Specifies the extended interrupt identifier to be used. This is commonly done via a symbolic name that is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

An extended interrupt identifier specifies the following:

- Whether it is a receive (RX), a send (TX) or a management (MGMT) interrupt
- The connection identifier in the range `0 … 255` for which the interrupt occurs

The following extended interrupt identifiers are defined:

Extended Interrupt Identifier	Meaning
DSIOETH_RX_INT_0	RxData interrupt for connection identifier 0.
…	…
DSIOETH_RX_INT_255	RxData interrupt for connection identifier 255.
DSIOETH_TX_INT_0	TxSent interrupt for connection identifier 0.
…	…
DSIOETH_TX_INT_255	TxSent interrupt for connection identifier 255.
DSIOETH_MGMT_INT_0	Management interrupt for connection identifier 0.
…	…
DSIOETH_MGMT_INT_255	Management interrupt for connection identifier 255.

IntHandler Specifies the address of the interrupt handler routine to be installed. You must implement this function before you can install it as an interrupt handler routine.

Return value

This function returns the address of the formerly installed interrupt handler.

Example

The following example shows how to implement and install an interrupt handler routine for the RxData interrupt at the socket with the connection identifier 0.

```
…
static void myHandleRecvInt0(void) {
    /* your interrupt handler routine */
    …
}

int main (void) {
    …
    /* install your interrupt handler */
    DsIoEth_InstallIntHandler(DSIOETH_RX_INT_0, myHandleRecvInt0);
    …
}
```

Related topics

References

Functions for Mgmt Interrupts

Purpose

To get information on the use of functions to handle Ethernet management events.

Mgmt interrupts occur, when the socket state of a connection changes. Every Mgmt interrupt occurrence is associated with a management event that specifies in which way the state changed.

Where to go from here

Information in this section

DsIoEth_acknowledgeMgmtInt

Syntax

```
void DsIoEth_acknowledgeMgmtInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To acknowledge a management interrupt.

Description

Use this function to acknowledge the management interrupt for the specified socket connection.

This function is relevant only if the socket of the connection has been created with the `DSIOETH_FLAG_IRQ_ACK_MODE` flag set.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_disableMgmtInt

Syntax

```
void DsIoEth_disableMgmtInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To disable a management interrupt.

Description

Use this function to disable the management interrupt for the specified socket connection.

After this call is executed, there are no more calls for the interrupt handler routine of the management interrupt for the specified socket connection. You can use **`DsIoEth_enableMgmtInt`** to re-enable the handling of this interrupt.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_enableMgmtInt

Syntax

```
void DsIoEth_enableMgmtInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To enable a management interrupt.

Description

Use this function to enable the management interrupt for the specified socket connection.

After the execution of this call, the interrupt handler routine for the management interrupt is executed if the interrupt occurs for the specified socket connection.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_getMgmtEvent

Syntax

```
UInt32 DsIoEth_getMgmtEvent(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To get the next received management event.

Description

Use this function to poll the latest management event that occurred for the specified socket connection. An element of the `DsEIoEthMgmtEvent`

enumeration is returned, which is defined in `<RCP_HIL_InstallationPath>\<BoardName>\Include\DsIoEth_Def.h`.

This function is commonly called if a management interrupt occurred.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

This function returns the following management events:

Management Event	Meaning
`0`	No further management event is available
DSIOETH_MGMT_EVENT_IDLE	An idle event was received after **`DsIoEth_trigger`** was called.
DSIOETH_MGMT_EVENT_PORT_OPEN	A port was opened using **`DsIoEth_open`**.
DSIOETH_MGMT_EVENT_PORT_CLOSED	A port was closed using **`DsIoEth_close`** or a TCP connection was closed by the remote station.
DSIOETH_MGMT_EVENT_PORT_CLOSED_ACK	**`DsIoEth_close`** was called, but the port was already closed.
DSIOETH_MGMT_EVENT_PORT_CONNECTED	A client or server connection has been established.
DSIOETH_MGMT_EVENT_PORT_ERROR	An error occurred, e.g., after **`DsIoEth_connect`** was called, but the server is not reachable.

Related topics

References

DsIoEth_trigger..... 60

DsIoEth_getMgmtEventCount

Syntax

```
Int32 DsIoEth_getMgmtEventCount(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To get the number of pending management events.

Description

Use this function to get the number of pending management events that you can examine using **`DsIoEth_getMgmtEvent`**.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

This function returns the following values:

Returned Value	Meaning
The number of pending management events.	The function was successfully executed.
`-1`	The function terminated with an error.

Related topics

References

DsIoEth_getMgmtEvent..... 58

DsIoEth_trigger

Syntax

```
Int32 DsIoEth_trigger(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To generate an idle event.

Description

Use this function to send an idle event to the specified socket connection and to generate a corresponding management interrupt. A call of **`DsIoEth_getMgmtEvent`** hereafter will return `DSIOETH_MGMT_EVENT_IDLE`.

This function is useful to keep state machines running if a socket connection has been closed.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

This function returns the following values:

Returned Value	Meaning
`0`	The function was executed successfully.
`-1`	The function terminated with an error.

Related topics

References

Functions for RxData Interrupts

Purpose

To get information on the use of RxData interrupt control functions.

RxData interrupts occur, when the network interface received data in its internal transmit buffer that is ready to be read.

Where to go from here

Information in this section

DsIoEth_acknowledgeRxDataInt

Syntax

```
void DsIoEth_acknowledgeRxDataInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To acknowledge an RxData receive interrupt.

Description

Use this function to acknowledge the RxData interrupt for the specified socket connection.

This function is relevant only, if the socket of the connection has been created with the `DSIOETH_FLAG_IRQ_ACK_MODE` flag set.

Parameters

ConnectionId	Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_disableRxDataInt

Syntax

```
void DsIoEth_disableRxDataInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To disable an RxData interrupt.

Description

Use this function to disable the RxData interrupt for the specified socket connection.

After this call is executed, there are no more calls for the interrupt handler routine of the receive interrupt for the specified socket connection. You can use **`DsIoEth_enableRxDataInt`** to re-enable the handling of this interrupt.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_enableRxDataInt

Syntax

```
void DsIoEth_enableRxDataInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To enable an RxData interrupt.

Description

Use this function to enable the RxData interrupt for the specified socket connection.

After the execution of this call, the interrupt handler routine for the receive interrupt is executed if the interrupt occurs for the specified socket connection.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_acknowledgeRxDataInt 62
DsIoEth_disableRxDataInt 63
DsIoEth_enableInt 52

Functions for TxSent Interrupts

Purpose

To get information on the use of TxSent interrupt control functions.

TxSent interrupts occur, when the network interface has sent data from the internal buffer to its destination and the buffer is ready to be written again.

Where to go from here

Information in this section

DsIoEth_acknowledgeTxSentInt

Syntax

```
void DsIoEth_acknowledgeTxSentInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To acknowledge a TxSent interrupt.

Description

Use this function to acknowledge the TxSent interrupt for the specified socket connection.

This function is relevant only if the socket of the used connection has been created with the `DSIOETH_FLAG_IRQ_ACK_MODE` flag set.

Parameters

ConnectionId	Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_disableTxSentInt

Syntax

```
void DsIoEth_disableTxSentInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To disable a TxSent interrupt.

Description

Use this function to disable the TxSent interrupt for the specified socket connection.

After this call is executed, there are no more calls for the interrupt handler routine of the sent interrupt for the specified socket connection. You can use **`DsIoEth_enableTxSentInt`** to re-enable the handling of this interrupt.

Parameters

ConnectionId	Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_enableTxSentInt

Syntax

```
void DsIoEth_enableTxSentInt(UInt32 ConnectionId)
```

Include file

`DsIoEth.h`

Purpose

To enable a TxSent interrupt.

Description

Use this function to enable the TxSent interrupt for the specified socket connection.

After the execution of this call, the interrupt handler routine for the sent interrupt is executed if the interrupt occurs for the specified socket connection.

Parameters

ConnectionId Specifies the unique connection identifier to be used.

Return value

None

Related topics

References

DsIoEth_acknowledgeTxSentInt 65
DsIoEth_disableTxSentInt 66
DsIoEth_enableMgmtInt 58

Conversion Functions

Purpose

To get information on the provided conversion functions.

Where to go from here

Information in this section

DsIoEth_htonl

Syntax

```
UInt32 DsIoEth_htonl(UInt32 HostLong)
```

Include file

`DsIoEth.h`

Purpose

To convert a 32-bit value from host byte order to network byte order, i.e., to big endian.

Description

This function is commonly used to convert a 32-bit IPv4 address to network byte order before it is assigned to a `DsSSockAddrIn` socket address as defined in `DsIoEth_SockAddr.h`.

Parameters

HostLong Specifies the 32-bit value to be converted in host byte order.

Return value

This function returns the converted value in network byte order.

Related topics

References

DsIoEth_ntohl.....71

DsIoEth_htons

Syntax

```
UInt16 DsIoEth_htons(UInt16 HostShort)
```

Include file

`DsIoEth.h`

Purpose

To convert a 16-bit value from host byte order to network byte order, i.e., to big endian.

Description

This function is commonly used to convert a 16-bit port number to network byte order before it is assigned to a `DsSSockAddrIn` socket address as defined in `DsIoEth_SockAddr.h`.

Parameters

HostShort Specifies the 16-bit value to be converted in host byte order.

Return value

This function returns the converted value in network byte order.

Related topics

References

DsIoEth_ntohs.....72

DsIoEth_inet_addr

Syntax

```
UInt32 DsIoEth_inet_addr(const char *strIpAddr)
```

Include file

`DsIoEth.h`

Purpose

To convert a string containing an IPv4 address to a 32-bit IP address in network byte order, i.e., to big endian.

Description

This function is commonly used to convert a string containing an IPv4 address to a IP address in network byte order before it is assigned to a `DsSSockAddrIn` socket address as defined in `DsIoEth_SockAddr.h`.

Parameters

strIpAddr Specifies the address of the string that contains an IPv4 address.

Return value

This function returns the following values:

Returned Value	Meaning
An IPv4 address in network-byte order.	The converted IP address.
`DSIOETH_INADDR_NONE`	The IP address can not be converted.

Related topics

References

DsIoEth_htonl.....69
DsIoEth_htons.....70

DsIoEth_ntohl

Syntax

```
UInt32 DsIoEth_ntohl(UInt32 NetLong)
```

Include file

`DsIoEth.h`

Purpose

To convert a 32-bit value from network byte order to host byte order, i.e., to little endian.

Description

This function is commonly used to convert a 32 bit IPv4 address to host byte order after reading it from a `DsSSockAddrIn` socket address as defined in `DsIoEth_SockAddr.h`.

Parameters

NetLong Specifies the 32-bit value to be converted in network byte order.

Return value

This function returns the converted value in host byte order.

Related topics

References

DsIoEth_htonl..... 69

DsIoEth_ntohs

Syntax

```
UInt16 DsIoEth_ntohs(UInt16 NetShort)
```

Include file

`DsIoEth.h`

Purpose

To convert a 16-bit value from network byte order to host byte order, i.e., to little endian.

Description

This function is commonly used to convert a 16-bit port number to host byte order after reading it from a `DsSSockAddrIn` socket address as defined in `DsIoEth_SockAddr.h`.

Parameters

NetShort Specifies the 16-bit value to be converted in network byte order.

Return value

This function returns the converted value in host-byte order.

Related topics

References

DsIoEth_htons..... 70

C

D

L