DSETH Ethernet Interface

RTLib Reference

Release 2021-A – May 2021

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

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2009 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

About This Reference

Contents

The DSETH Real-Time Library (RTLib) provides the C functions and macros you need to program the DS867 LVDS-Ethernet interface or the ETH_TP1 interface of a MicroAutoBox II.

Supported Hardware

The following dSPACE systems are supported:

- PHS-bus-based system with a DS1006 processor board and one or more DS4121 ECU Interface Boards
- MicroAutoBox II

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

Symbol	Description
📖	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.

Data Type Definitions

Where to go from here

Information in this section

Predefined Symbols

Introduction

The following tables list the defines, which are predefined in the `dseth.h`, `dseth_custom_ds867.h` and `dseth_custom_eth_tp1.h` include files.

Debug level flag defines

DSEth Debug Level Flag	Meaning
DSETH_OBJ_INIT_DEBUG_LEVEL_DISABLED	Disable additional debug information. No additional debug messages are generated.
DSETH_OBJ_INIT_DEBUG_LEVEL_DEVICE_INFO	Shows device information of the connected device. The extended information is shown after the device is initialized.

Object bandwidth defines

Symbol	Value	Meaning
DSETH_OBJ_BANDWIDTH_MAX_DEFAULT	DSETH_OBJ_INIT_BANDWIDTH_MAX_100MBit	Default maximum bandwidth

Object initialization parameter defines

Symbol	Value	Meaning
DSETH_OBJ_INIT_BANDWIDTH_MAX_DEFAULT	0	Use the device's default as bandwidth limitation (normally maximum available speed).
DSETH_OBJ_INIT_BANDWIDTH_MAX_100MBit	100	Maximum bandwidth is 100 MBit/s.
DSETH_OBJ_INIT_BANDWIDTH_MAX_1GBit	1000	Maximum bandwidth is 1 GBit/s.

RX data buffer defines

Symbol	Value	Meaning
DSETH_RX_BUFFER_COUNT_DEFAULT	100	Default number of RX buffer elements
DSETH_RX_BUFFER_SIZE_DEFAULT	1500	Default size of RX data buffer in bytes

TX data buffer defines

Symbol	Value	Meaning
DSETH_TX_BUFFER_COUNT_DEFAULT	50	Default number of TX buffer elements
DSETH_TX_BUFFER_SIZE_DEFAULT	1500	Default size of TX data buffer in bytes

TX state defines

Symbol	Value	Meaning
DSETH_TX_STATE_ALLOCATED	0x01	TX data buffer is newly allocated. Sending is possible.
DSETH_TX_STATE_QUEUED	0x02	The TX data buffer is queued for sending. Sending will be processed as soon as possible.
DSETH_TX_STATE_PROCESSING	0x03	The TX data buffer has been written to the interface's TX queue. Waiting for the send finish status.
DSETH_TX_STATE_SUCCESSED	0x04	Sending of data buffer was successful.
DSETH_TX_STATE_FAILED_ERROR	0x05	Sending of data buffer caused an interface-specific send error (in most cases due to an unresolvable IP address).
DSETH_TX_STATE_FAILED_TIMEOUT	0x06	Sending of TX data buffer failed due to a timeout. The message could not be sent in the time DsEthSSocketInit::TimeOutTX.
DSETH_TX_STATE_DELETED	0x07	TX data buffer was deleted.

TX defines

Symbol	Value	Meaning
DSETH_TX_TIMEOUT	0.0005	Default TX timeout in seconds

Default initialization defines

Symbol	Value	Meaning

DSETH_OBJ_INIT_DEFAULT	{
	0 /* Version */,
	NULL /* pComHandle */,
	DSETH_RX_BUFFER_SIZE_DEFAULT /* RXBufferSize */,
	DSETH_RX_BUFFER_COUNT_DEFAULT /* RXBufferCount */,
	DSETH_TX_BUFFER_SIZE_DEFAULT /* TXBufferSize */,
	DSETH_TX_BUFFER_COUNT_DEFAULT /* TXBufferCount */,
	NULL /* pSpecific */,
	NULL /* pCallback */,
	0x0 /* DebugLevel */,
	DSETH_OBJ_BANDWIDTH_MAX_DEFAULT /* BandwidthMax */
	}
		Default values for the `DsEthSObjInit` struct. It is strongly recommended to always use the macro to initialize objects of this type.
		For an example, refer to DsEthSObjInit on page 18.

DSETH_SOCKET_INIT_DEFAULT	{
	0 /* Version */,
	DSETH_ACCESS_SOCKET_DISABLED /* ConfigFlags */,
	0 /* IPLocal */,
	0 /* IPRemote */,
	0 /* PortLocal */,
	0 /* PortRemote */,
	NULL /* pSpecific */,
	NULL /* pCallback */,
	DSETH_TX_TIMEOUT /* TimeOutTX */,
	0 /* Priority */,
	0 /* TXRetryCount */,
	0 /* InterpacketGap */
	}
		Default values for the `DsEthSSocketInit` struct. It is strongly recommended to always use the macro to initialize objects of this type.
		For an example, refer to DsEthSSocketInit on page 20.

DSETH_CSTM_DS867_COM_HANDLE_INIT_DEFAULT	{
	0 /* Version */,
	0 /* Base */,
	0 /* Channel */
	}
		Default values for the `DsEthCstmDS867SComHandle` struct. It is strongly recommended to always use the macro to initialize objects of this type.
		For an example, refer to DsEthCstmDS867SComHandle on page 11.

Symbol	Value	Meaning

DSETH_CSTM_ETH_TP1_COM_HANDLE_INIT_DEFAULT	{
	0 /* Version */,
	0 /* ModuleNo */
	}
		Default values for the `DsEthCstmEthTp1SComHandle` struct. It is strongly recommended to always use the macro to initialize objects of this type.
		For an example, refer to DsEthCstmEthTp1SComHandle on page 12.

DsEthCstmDS867SComHandle

Introduction

The `DsEthCstmDS867SComHandle` structure is used for DSETH custom communication object initialization if an LVDS-Ethernet link cable (DS867) is used.

Note

To ensure API version compatibility, it is recommended to use the `DSETH_CSTM_DS867_COM_HANDLE_INIT_DEFAULT` macro for `DsEthCstmDS867SComHandle` structure initialization. For information on its default values, refer to Predefined Symbols on page 8.

Syntax

```
typedef struct tagDsEthCstmDS867AccessComHandle
      DsEthCstmDS867SComHandle;
struct tagDsEthCstmDS867AccessComHandle{
   UInt32  Version;
   UInt32  Base;
   UInt32  Channel;
};
```

Include file

```
dseth_custom_ds867.h
```

Members

Version Version number of the structure. The current version number is 0.

Base PHS-bus base address of the connected DS4121 board or number of the connected ECU module (MicroAutoBox II) that provides communication.

Channel Channel number of the connected board to be used. This member is used in combination with a DS4121 board only.

Example

The following example shows how to initialize a DSETH custom communication object if an LVDS-Ethernet link cable is used:

```
{
  pDsEthCfgSComHandle pComHandle;
  phs_addr_t board_base;
  ...

  /* get DS4121 base address of first board */
  board_base = get_peripheral_addr(DS4121_BOARD_ID, 1 /* board number */);

  /* initialize DS4121 board */
  ds4121_init(board_base);

  /* Creating generic communication handler for DS867 */
  {
    DsEthCstmDS867SComHandle comHandleDS867 =
            DSETH_CSTM_DS867_COM_HANDLE_INIT_DEFAULT;

    comHandleDS867.Base = board_base;
    comHandleDS867.Channel = 1;
    pComHandle = dsEthCstmDS867( &comHandleDS867 );
  }
  ...
}
```

Related topics

Basics

Predefined Symbols.. 8

DsEthCstmEthTp1SComHandle

Introduction

The `DsEthCstmEthTp1SComHandle` structure is used for DSETH custom communication object initialization using an ETH_TP1 interface.

Note

To ensure API version compatibility, it is recommended to use the `DSETH_CSTM_ETH_TP1_COM_HANDLE_INIT_DEFAULT` macro for `DsEthCstmEthTp1SComHandle` structure initialization. For information on its default values, refer to Predefined Symbols on page 8.

Syntax

```
typedef struct tagDsEthCstmEthTp1AccessComHandle
      DsEthCstmEthTp1SComHandle;
struct tagDsEthCstmEthTp1AccessComHandle{
   UInt32  Version;
   UInt32  ModuleNo;
};
```

Include file

```
dseth_custom_eth_tp1.h
```

Members

Version Version number of the structure. The current version number is 0.

ModuleNo Number of the connected ECU module that provides communication.

Example

The following example shows how to initialize a DSETH custom communication object if an ETH_TP1 interface is used:

```
{
  pDsEthCfgSComHandle pComHandle;
  ...

  /* Creating generic communication handler object for ETH_TP1 interface */
  {
    dsEthCstmEthTp1SComHandle  comHandleEthTp1 =
            DSETH_CSTM_ETH_TP1_COM_HANDLE_INIT_DEFAULT;
    comHandleEthTp1.ModuleNo = 1; /* Index of ETH_TP1 module */
    pComHandle = dsEthCstmEthTp1( &comHandleEthTp1 );
  }
  ...
}
```

Related topics

Basics

Predefined Symbols.. 8

DsEthSDataBufferObj

Introduction

The `DsEthSDataBufferObj` structure contains information on DSETH data buffer objects.

The `DsEthSDataBufferObj` structure is a forward declaration for the `tagDsEthSDataBufferObj` structure. `tagDsEthSDataBufferObj` is the object structure defining a DSETH data buffer object. Each instance identifies a specific data buffer.

Syntax

```
typedef struct tagDsEthSDataBufferObj DsEthSDataBufferObj;
struct tagDsEthSDataBufferObj{
   DSETHUInt32  Version;
   DSETHUInt8  *pData;
   DSETHUInt32  SizeMax;
   DSETHUInt32  Size;
   DSETHUInt32  Flags;
   DSETHUInt16  Handle;
   DSETHTimeStamp  TimeStamp;
   DSETHTimeStamp  TimeOut;
   pDsEthSSocketObj  pSocketObj;
   pDsEthSObj  pDsEthObj;
   pDsEthAccessSBufferControlTX  pDsEthAccessBufferControlTX;
   pDsEthAccessSBufferControlRX  pDsEthAccessBufferControlRX;
   pDsEthSDataBufferObj  pNext;
   pDsEthSDataBufferObj  pBefore;
   void *  pSpecific;
   DSETHUInt16  TXRetryCount;
   DSETHUInt32  TXState;
};
```

Include file

```
dseth.h
```

Members

Version Version number of the structure. The current version number is 0. This member is implicitly handled by the buffer handling functions and must not be changed.

pData Pointer to the data buffer memory.

SizeMax Maximum size of the data buffer (in bytes). This member is implicitly handled by the buffer handling functions and must not be changed.

Size Currently used data buffer size (in bytes).

Flags Flags indicating how data sending is processed. This member is implicitly handled by the buffer handling functions and must not be changed.

Handle Internal data buffer handle that the data of this buffer was sent with. This member is implicitly handled by the buffer handling functions and must not be changed.

TimeStamp Time (in seconds) the data buffer was sent (TX data buffer) or received (RX data buffer). This member is implicitly handled by the buffer handling functions and must not be changed.

TimeOut Timeout for sending the data buffer. The default is the DsEthSSocketInit::TimeOutTX member value. This member is implicitly handled by the buffer handling functions and must not be changed.

pSocketObj Pointer to the related socket object. This member is implicitly handled by the buffer handling functions and must not be changed.

pDsEthObj Pointer to the related DSETH object. This member is implicitly handled by the buffer handling functions and must not be changed.

pDsEthAccessBufferControlTX Pointer to TX buffer control struct. NULL, if RX buffer only. This member is implicitly handled by the buffer handling functions and must not be changed.

pDsEthAccessBufferControlRX Pointer to RX buffer control struct. NULL, if RX buffer only. This member is implicitly handled by the buffer handling functions and must not be changed.

pNext Pointer to the next data buffer object. NULL terminates the chain. This member is implicitly handled by the buffer handling functions and must not be changed.

pBefore Pointer to the previous data buffer object. NULL terminates the chain. This member is implicitly handled by the buffer handling functions and must not be changed.

pSpecific Pointer to top-level-specific information. The default is the DsEthSSocketInit::pSpecific member value. Use this pointer to make references to specific top-level information.

TXRetryCount Number of remaining retries if the sending of this buffer failed. This value is independent of the TimeOut member value, i.e., if a timeout occurs the message is not sent any longer even if the TXRetryCount still indicates further retries. The default is the DsEthSSocketInit::TXRetryCount value.

TXState Current TX state. This member is implicitly handled by the buffer handling functions and must not be changed.
For information on the possible TX states, refer to Predefined Symbols on page 8.

Related topics

Basics

Predefined Symbols.. 8

DsEthSObj

Introduction

`DsEthSObj` contains information on DSETH communication objects.

The `DsEthSObj` structure is a forward declaration for the `tagDsEthSObj` structure. `tagDsEthSObj` is the object structure used to define DSETH interfaces. Each instance identifies a specific DSETH interface.

Note

The members of the `DsEthSObj` structure are implicitly handled by the DSETH API functions and must not be changed explicitly.

Syntax

```
typedef struct tagDsEthSObj DsEthSObj;
struct tagDsEthSObj{
   DSETHUInt8  * pName;
   DSETHUInt32  InitRequired;
   pDsEthAccessSObj  pAccessObj;
   DsEthSObjInit  ObjInit;
   pDsEthSSocketObj  pSocketObjChainBegin;
   pDsEthSSocketObj  pSocketObjChainEnd;
   pDsEthSSocketObj  * ppSocketObjIDReferenceDB;
   DSETHUInt32  SocketObjIDReferenceDBSize;
   pDsEthSDataBufferObj  pRxDataBufferObjRXFreeChain;
   pDsEthSDataBufferObj  pTxDataBufferObjTXFreeChain;
   pDsEthSDataBufferObj  pTxDataBufferObjTXSendChainBegin;
   pDsEthSDataBufferObj  pTxDataBufferObjTXSendChainEnd;
   pDsEthSDataBufferObj  pTxDataBufferObjTXCheckChainBegin;
   pDsEthSDataBufferObj  pTxDataBufferObjTXCheckChainEnd;
   DSETHUInt32  TXFreeChainCount;
   DSETHUInt32  TXSendChainCount;
   DSETHUInt32  TXCheckChainCount;
   pDsEthSObj  pNext;
   void * pSpecific;
   DSETHUInt16  SocketID;
   DSETHUInt32  ConnectionStateLast;
   DSETHUInt32  ConnectionStatePrepare;
   DSETHTimeStamp  ConnectionStateChangeTime;
   DSETHTimeStamp  ConnectionStateChangeTimeout;
   DSETHUInt16  TXSendOverrun;
   DSETHUInt16  TXCheckOverrun;
   DSETHUInt32  DebugLevel;
};
```

Include file

```
dseth.h
```

Members

pName Name of the object.

InitRequired Flag indicating whether (re)initialization is required.

pAccessObj Pointer to quick access-specific communication handle.

ObjInit Copy of the initialization parameters.

pSocketObjChainBegin Start of the socket object chain.

pSocketObjChainEnd End of the socket object chain.

ppSocketObjIDReferenceDB Reference list to get the socket object belonging to a socketID.

SocketObjIDReferenceDBSize Number of socket reference entries in the ppSocketObjIDReferenceDB list.

pRxDataBufferObjRXFreeChain Start of the chain of free RX data buffer objects.

pTxDataBufferObjTXFreeChain Start of the chain of free TX data buffer objects.

pTxDataBufferObjTXSendChainBegin Start of the chain of TX data buffer objects to be sent.

pTxDataBufferObjTXSendChainEnd End of the chain of TX data buffer objects to be sent.

pTxDataBufferObjTXCheckChainBegin Start of the chain of TX data buffer objects to be checked.

pTxDataBufferObjTXCheckChainEnd End of the chain of TX data buffer objects to be checked.

TXFreeChainCount Number of data buffer objects in the chain of free TX data buffer objects.

TXSendChainCount Number of data buffer objects in the chain of TX data buffer objects to be sent.

TXCheckChainCount Number of data buffer objects in the chain of TX data buffer objects to be checked.

pNext Next object in the DSETH object chain.

pSpecific Pointer to top level-specific information. This value is set at initialization time.

SocketID Internally used member.

ConnectionStateLast Internally used member.

ConnectionStatePrepare Internally used member.

ConnectionStateChangeTime Internally used member.

ConnectionStateChangeTimeout Internally used member.

TXSendOverrun Internally used member.

TXCheckOverrun Internally used member.

DebugLevel Flag indicating whether additional debug information for the connected device is to be displayed in the log file.

Related topics

Basics

Predefined Symbols.. 8

DsEthSObjInit

Introduction

The `DsEthSObjInit` structure is used for DSETH communication object initialization.

Note

To ensure API version compatibility, it is recommended to use the `DSETH_OBJ_INIT_DEFAULT` macro for `DsEthSObjInit` structure initialization. For information on its default values, refer to Predefined Symbols on page 8.

Syntax

```
typedef struct tagDsEthSObjInit DsEthSObjInit;
struct tagDsEthSObjInit{
   DSETHUInt32  Version;
   pDsEthAccessComHandle  pComHandle;
   DSETHUInt32  RXBufferSize;
   DSETHUInt32  RXBufferCount;
   DSETHUInt32  TXBufferSize;
   DSETHUInt32  TXBufferCount;
   void  * pSpecific;
   void  (* pCallback)(pDsEthSObj pDsEthObj);
   DSETHUInt32  DebugLevel;
   DSETHUInt32  BandwidthMax;
};
```

Include file

```
dseth.h
```

Members

Version Version number of the structure. The current version number is 0.

pComHandle Pointer to specific communication handler. The handler is used by the dseth custom layer to communicate with a specific device.

The communication handler is generated by one of the following functions:

- `dsEthCstmDS867`, if the LVDS-Ethernet cable (DS867) is used
- `dsEthCstmEthTp1`, if the ETH_TP1 interface of a MicroAutoBox II is used

RXBufferSize Maximum number of payload bytes in each RX data buffer

RXBufferCount Number of RX buffers

TXBufferSize Maximum number of payload bytes in each TX data buffer

TXBufferCount Number of TX buffers

pSpecific Optional pointer to specific information. The pointer is not used by the DSETH, but is available in the pDsEthSObj object. Use this pointer to refer to your own data structures and information as top-level information related to the current DSETH interface.

pCallback Callback function pointer. If defined, the function is called each time new data arrived for the DSETH interface. NULL disables the function.

- Parameter of the Callback function: `pDsEthObj`

  pDsEthObj is the pointer to the pDsEthObj the data was received with. Use the pSpecific pointer to refer to a related top-level information of your own structure.
- Return value of the Callback function: None

DebugLevel Debug level information. Choose one or OR-mask several of the available flags.

For information on the possible debug flags, refer to Debug level flag defines (see Predefined Symbols on page 8).

BandwidthMax Maximum interface bandwidth.

For information on the possible maximum bandwidth values, refer to Object bandwidth defines (see Predefined Symbols on page 8).

Note

The actually used bandwidth is determined by the Ethernet bus the interface is connected to and the maximum bandwidth supported by the Ethernet interface. If you specify a bandwidth that is not supported by the Ethernet interface, the interface implicitly reduces the bandwidth to the maximum supported bandwidth.

Example

The following example shows how to initialize a DSETH communication object:

```
{
  DsEthSObjInit ObjInit = DSETH_OBJ_INIT_DEFAULT;
  pDsEthSObj pDsEthObj = NULL;
  ...
  pDsEthObj = dsEthObjInit("MyEthernetInterfaceName",
                           &ObjInit);
}
```

Related topics

Basics

DsEthSSocketInit

Introduction

The `DsEthSSocketInit` structure is used for DSETH socket object initialization.

Note

To ensure API version compatibility, it is recommended to use the `DSETH_SOCKET_INIT_DEFAULT` macro for `DsEthSSocketInit` structure initialization. For information on its default values, refer to Predefined Symbols on page 8.

Syntax

```
typedef struct tagDsEthSSocketInit DsEthSSocketInit;
struct tagDsEthSSocketInit{
   DSETHUInt32  Version;
   DSETHUInt32  ConfigFlags;
   DSETHUInt32  IPLocal;
   DSETHUInt32  IPRemote;
   DSETHUInt16  PortLocal;
   DSETHUInt16  PortRemote;
   void  * pSpecific;
   void  (* pCallback)(pDsEthSSocketObj  pSocketObj);
   DSETHTimeStamp  TimeOutTX;
   DSETHUInt32  Priority;
   DSETHUInt16  TXRetryCount;
   DSETHUInt16  InterpacketGap;
};
```

Include file

```
dseth.h
```

Members

Version Version number of the structure. The current version number is 0.

ConfigFlags Configuration flags, providing information on the socket direction (RX socket, TX socket, bidirectional socket) and the access mode.

IPLocal Local IP address. Use the `DSETHIPADDR` macro to convert the data format of a common IP address.

Example: `SocketInit.IPLocal = DSETHIPADDR(192, 168, 0, 1)`

IPRemote IP address of the remote system. Use the `DSETHIPADDR` macro to convert the data format of a common IP address.

Example: `SocketInit.IPRemote = DSETHIPADDR(192, 168, 0, 10)`

PortLocal Local port address. The local port address describes the port number incoming packets are received and outgoing packets are sent with (source port) using the current socket.

PortRemote Port number of the remote system. This member describes the port number of the remote system incoming packets are received to and outgoing packets are sent from (destination port) using the current socket.

pSpecific Optional pointer to specific information. The pointer is not used by the DSETH, but is available in the pDsEthSObj object. Use this pointer to refer to your own data structures and information as top-level information related to the current socket.

pCallback Callback function pointer. If defined, the function is called each time new data arrived for the related socket. NULL disables the function.

- Parameter of the Callback function: `pSocketObj`

  pSocketObj is the pointer to the pDsEthSocketObj the data was received with. If required, use the pSpecific pointer to refer to top-level information related to this socket.
- Return value of the Callback function: None

TimeOutTX Timeout (in seconds) for TX messages that cannot be sent successfully. Messages are no longer sent if the interface was not able to send this message in the configured period.

Priority Socket priority. The higher the value, the lower the priority. When new data arrives, data assignment is performed according to the socket priority. This is important if there are several sockets matching an incoming data packet. In that case, the resulting data buffer is assigned to the matching socket with the highest priority.

TXRetryCount Number of retries of sending TX messages

InterpacketGap Time interval in seconds, used as the minimum delay between two messages that are sent. Negative values are not allowed.

Example

The following example shows how to initialize a DSETH socket object:

```
{
  DsEthSSocketInit  SocketInit = DSETH_SOCKET_INIT_DEFAULT;
  pDsEthSSocketObj  pSocketObj = NULL;
  ...
  pSocketObj = dsEthSocketObjInit(pDsEthObj, &SocketInit);
}
```

Related topics

Basics

Predefined Symbols..8

DsEthSSocketObj

Introduction

The `DsEthSSocketObj` structure contains information on DSETH socket objects.

The `DsEthSSocketObj` structure is a forward declaration for the `tagDsEthSSocketObj` structure. `tagDsEthSSocketObj` is the object structure used to define DSETH sockets. Each instance identifies a specific socket.

Note

The members of the `DsEthSSocketObj` structure are implicitly handled by the DSETH API functions and must not be changed explicitly.

Syntax

```
typedef struct tagDsEthSSocketObj DsEthSSocketObj;
struct tagDsEthSSocketObj{
   DSETHUInt32  Version;
   DSETHUInt16  SocketID;
   DSETHUInt16  SocketNo;
   DSETHUInt32  Counter;
   DsEthSSocketInit  SocketInit;
   DsEthAccessSConfigSocket  ConfigSocket;
   pDsEthSObj  pDsEthObj;
   pDsEthSSocketObj  pNext;
   pDsEthSSocketObj  pBefore;
   pDsEthSDataBufferObj  DataBufferObjRXChainBegin;
   pDsEthSDataBufferObj  DataBufferObjRXChainEnd;
   void  * pSpecific;
   DSETHUInt32  CallbackProcess;
};
```

Include file

```
dseth.h
```

Members

Version Version number of the structure. The current version number is 0.

SocketID Socket identifier, used to identify the current socket.

SocketNo Socket number, specifies the position in the socket list the socket is saved to. The lower a socket number, the higher the priority of the socket.

Counter TX buffer send counter

SocketInit Copy of the socket initialization parameters

ConfigSocket DSETH access socket configuration parameters

pDsEthObj Pointer to the related DSETH object

pNext Pointer to the next socket element. NULL terminates the chain.

pBefore Pointer to the previous socket element. NULL terminates the chain.

DataBufferObjRXChainBegin Start of the RX data buffer object chain

DataBufferObjRXChainEnd End of the RX data buffer object chain

pSpecific Pointer to top-level-specific parameters. Use this parameter to refer to specific data structures and information as top-level information related to the current socket. The value will be set to the DsEthSSocketInit::pSpecific member value at socket initialization time.

CallbackProcess Flag indicating to process callback for this socket if data was received. This parameter is implicitly handled within the DSETH API functions.

Related topics

Basics

Predefined Symbols.. 8

pDsEthCstmDS867SComHandle

Introduction

`pDsEthCstmDS867SComHandle` is a pointer to the `DsEthCstmDS867SComHandle` structure.

Syntax

```
typedef struct tagDsEthCstmDS867AccessComHandle*
pDsEthCstmDS867SComHandle;
struct tagDsEthCstmDS867AccessComHandle{
   UInt32  Version;
   UInt32  Base;
   UInt32  Channel
};
```

Include file

```
dseth_custom_ds867.h
```

Members

For information on the members of the **`pDsEthCstmDS867SComHandle`** structure, refer to DsEthCstmDS867SComHandle on page 11.

Related topics

References

DsEthCstmDS867SComHandle.. 11

pDsEthCstmEthTp1SComHandle

Introduction

`pDsEthCstmEthTp1SComHandle` is a pointer to the `DsEthCstmEthTp1SComHandle` structure.

Syntax

```
typedef struct tagDsEthCstmEthTp1AccessComHandle*
pDsEthCstmEthTp1SComHandle;
struct tagDsEthCstmEthTp1AccessComHandle{
   UInt32  Version;
   UInt32  ModuleNo
};
```

Include file

```
dseth_custom_eth_tp1.h
```

Members

For information on the members of the `pDsEthCstmEthTp1SComHandle` structure, refer to DsEthCstmEthTp1SComHandle on page 12.

Related topics

References

DsEthCstmEthTp1SComHandle.. 12

pDsEthSDataBufferObj

Introduction

`pDsEthSDataBufferObj` is a pointer to the `DsEthSDataBufferObj` structure.

Syntax

```
typedef struct tagDsEthSDataBufferObj* pDsEthSDataBufferObj;
struct tagDsEthSDataBufferObj{
   DSETHUInt32  Version;
   DSETHUInt8  *pData;
   DSETHUInt32  SizeMax;
   DSETHUInt32  Size;
   DSETHUInt32  Flags;
   DSETHUInt16  Handle;
   DSETHTimeStamp  TimeStamp;
   DSETHTimeStamp  TimeOut;
   pDsEthSSocketObj  pSocketObj;
   pDsEthSObj  pDsEthObj;
   pDsEthAccessSBufferControlTX  pDsEthAccessBufferControlTX;
   pDsEthAccessSBufferControlRX  pDsEthAccessBufferControlRX;
   pDsEthSDataBufferObj  pNext;
   pDsEthSDataBufferObj  pBefore;
   void *  pSpecific;
   DSETHUInt16  TXRetryCount;
   DSETHUInt32  TXState;
};
```

Include file

```
dseth.h
```

Members

For information on the members of the `pDsEthSDataBufferObj` structure, refer to DsEthSDataBufferObj on page 13.

Related topics

References

DsEthSDataBufferObj 13

pDsEthSObj

Introduction

`pDsEthSObj` is a pointer to the `DsEthSObj` structure.

Syntax

```
typedef struct tagDsEthSObj* pDsEthSObj;
struct tagDsEthSObj{
   DSETHUInt8  * pName;
   DSETHUInt32  InitRequired;
   pDsEthAccessSObj  pAccessObj;
   DsEthSObjInit  ObjInit;
   pDsEthSSocketObj  pSocketObjChainBegin;
   pDsEthSSocketObj  pSocketObjChainEnd;
   pDsEthSSocketObj  * ppSocketObjIDReferenceDB;
   DSETHUInt32  SocketObjIDReferenceDBSize;
   pDsEthSDataBufferObj  pRxDataBufferObjRXFreeChain;
   pDsEthSDataBufferObj  pTxDataBufferObjTXFreeChain;
   pDsEthSDataBufferObj  pTxDataBufferObjTXSendChainBegin;
   pDsEthSDataBufferObj  pTxDataBufferObjTXSendChainEnd;
   pDsEthSDataBufferObj  pTxDataBufferObjTXCheckChainBegin;
   pDsEthSDataBufferObj  pTxDataBufferObjTXCheckChainEnd;
   DSETHUInt32  TXFreeChainCount;
   DSETHUInt32  TXSendChainCount;
   DSETHUInt32  TXCheckChainCount;
   pDsEthSObj  pNext;
   void* pSpecific;
   DSETHUInt16  SocketID;
   DSETHUInt32  ConnectionStateLast;
   DSETHUInt32  ConnectionStatePrepare;
   DSETHTimeStamp  ConnectionStateChangeTime;
   DSETHTimeStamp  ConnectionStateChangeTimeout;
   DSETHUInt16  TXSendOverrun;
   DSETHUInt16  TXCheckOverrun;
   DSETHUInt32  DebugLevel
};
```

Include file

```
dseth.h
```

Members

For information on the members of the **`pDsEthSObj`** structure, refer to DsEthSObj on page 15.

Related topics

References

DsEthSObj.. 15

pDsEthSObjInit

Introduction

`pDsEthSObjInit` is a pointer to the `DsEthSObjInit` structure.

Syntax

```
typedef struct tagDsEthSObjInit* pDsEthSObjInit;
struct tagDsEthSObjInit{
   DSETHUInt32  Version;
   pDsEthAccessComHandle  pComHandle;
   DSETHUInt32  RXBufferSize;
   DSETHUInt32  RXBufferCount;
   DSETHUInt32  TXBufferSize;
   DSETHUInt32  TXBufferCount;
   void  * pSpecific;
   void  (* pCallback)(pDsEthSObj  pDsEthObj);
   DSETHUInt32  DebugLevel
   DSETHUInt32  BandwidthMax
};
```

Include file

```
dseth.h
```

Members

For information on the members of the `pDsEthSObjInit` structure, refer to DsEthSObjInit on page 18.

Related topics

References

DsEthSObjInit.. 18

pDsEthSpecific

Introduction

`pDsEthSpecific` is a type definition for a specific parameter structure. It is set to `void *` to append top-level data structures of various types to make custom references to data structures and information related to a dsEth object.

Syntax

```
typedef void* pDsEthSpecific
```

Include file

```
dseth.h
```

Members

None

pDsEthSSocketInit

Introduction

`pDsEthSSocketInit` is a pointer to the `DsEthSSocketInit` structure.

Syntax

```
typedef struct tagDsEthSSocketInit* pDsEthSSocketInit;
struct tagDsEthSSocketInit{
   DSETHUInt32  Version;
   DSETHUInt32  ConfigFlags;
   DSETHUInt32  IPLocal;
   DSETHUInt32  IPRemote;
   DSETHUInt16  PortLocal;
   DSETHUInt16  PortRemote;
   void  * pSpecific;
   void  (* pCallback)(pDsEthSSocketObj  pSocketObj);
   DSETHTimeStamp  TimeOutTX;
   DSETHUInt32  Priority;
   DSETHUInt16  TXRetryCount;
   DSETHUInt16  InterpacketGap;
};
```

Include file

```
dseth.h
```

Members

For information on the members of the `pDsEthSSocketInit` structure, refer to DsEthSSocketInit on page 20.

Related topics

References

pDsEthSSocketObj

Introduction

`pDsEthSSocketObj` is a pointer to the `DsEthSSocketObj` structure.

Syntax

```
typedef struct tagDsEthSSocketObj* pDsEthSSocketObj;
struct tagDsEthSSocketObj{
   DSETHUInt32  Version;
   DSETHUInt16  SocketID;
   DSETHUInt16  SocketNo;
   DSETHUInt32  Counter;
   DsEthSSocketInit  SocketInit;
   DsEthAccessSConfigSocket  ConfigSocket;
   pDsEthSObj  pDsEthObj;
   pDsEthSSocketObj  pNext;
   pDsEthSSocketObj  pBefore;
   pDsEthSDataBufferObj  DataBufferObjRXChainBegin;
   pDsEthSDataBufferObj  DataBufferObjRXChainEnd;
   void  * pSpecific;
   DSETHUInt32  CallbackProcess;
};
```

Include file

```
dseth.h
```

Members

For information on the members of the `pDsEthSSocketObj` structure, refer to DsEthSSocketObj on page 22.

Related topics

References

DsEthSSocketObj.. 22

Initialization and Controlling Functions

Introduction

To get information on the functions used to perform the initialization process and background executions, and to check the Ethernet connection.

Where to go from here

Information in this section

dsEthBackground

Syntax

```
void dsEthBackground()
```

Include file	`dseth.h`
Purpose	To process dsEth interface background executions.
Description	The `dsEthBackground` function must be called in your application's background task.
Parameters	None
Return value	None

dsEthConnectionCheck

Syntax	`DsEthTError dsEthConnectionCheck(pDsEthSObj pDsEthObj)`
Include file	`dseth.h`
Purpose	To check if a DSETH device is available and connected to the Ethernet.
Description	You can evaluate the return value to get information on whether the connection is currently established.
Parameters	**pDsEthObj** Pointer to the DSETH communication object
Return value	Returns the following error codes:

Error Code	Meaning
DSETH_ERR_SUCCESS	No error occurred during the operation. The connection between the DSETH device and the Ethernet is established.
DSETH_ERR_ILLEGAL	The pDsEthObj parameter is illegal (for example, NULL).
DSETH_ERR_COM_DISCONNECTED[1)]	Communication is not possible because the DSETH device is currently not connected.

Error Code	Meaning
DSETH_ERR_UNINITIALIZED	The DSETH device is physically connected, but is currently not initialized. Communication is not possible.
DSETH_ERR_ETH_DISCONNECTED	The DSETH device is physically connected and initialized, but there is no connection to the Ethernet.

1) Note: Not relevant for the ETH_TP1 device.

dsEthCstmDS867

Syntax

```
pDsEthCfgSComHandle dsEthCstmDS867(
      pDsEthCstmDS867SComHandle pDS867comHandle)
```

Include file

```
dseth_custom_ds867.h
```

Purpose

To create a communication handler for a dsEthCstmDS867 object.

Use this function, if the LVDS-Ethernet cable (DS867) is used.

Parameters

pDS867comHandle	Reference to a DS867 communication handle structure

Return value

This function returns a reference to a communication handle.

Related topics

References

DsEthSObjInit 18

dsEthCstmEthTp1

Syntax

```
pDsEthCfgSComHandle dsEthCstmEthTp1(
      pDsEthCstmEthTp1SComHandle pEthTp1comHandle)
```

Include file

```
dseth_custom_eth_tp1.h
```

Purpose	To create a communication handle for a dsEthCstmEthTp1 object. Use this function, if the ETH_TP1 interface of a MicroAutoBox II is used. The ETH_TP1 interface is a built-in ETH device of the MicroAutoBox II.
Parameters	**pEthTp1comHandle** Reference to a DS1401 communication handle structure
Return value	This function returns a reference to a communication handle.
Related topics	References DsEthSObjInit.. 18

dsEthObjInit

Syntax

```
pDsEthSObj dsEthObjInit(
      DSETHUInt8 *  pObjName,
      pDsEthSObjInit  pObjInit);
```

Include file

```
dseth.h
```

Purpose

To initialize a DSETH communication object. A DSETH communication object represents one physical DSETH interface.

Parameters

pObjName	Name of the object
pObjInit	Pointer to the communication object initialization parameter structure

Return value

This function returns:

- NULL, if the function failed.
- A pointer to a new DsEthSObj structure.

Related topics

Basics

dsEthSocketObjInit

Syntax

```
pDsEthSSocketObj dsEthSocketObjInit(
      pDsEthSObj  pDsEthObj,
      pDsEthSSocketInit  pSocketInit);
```

Include file

```
dseth.h
```

Purpose

To initialize a DSETH socket object.

Description

You can configure up to four sockets.

Parameters

pDsEthObj	Pointer to the DSETH communication object initialized using the `dsEthObjInit` function
pSocketInit	Pointer to the socket object initialization parameter structure

Return value

This function returns:

- NULL, if the function failed.
- A pointer to the new DsEthSSocketObj structure.

Related topics

Basics

dsEthSocketObjDelete

Syntax

```
DsEthTError dsEthSocketObjDelete(pDsEthSSocketObj pSocketObj)
```

Include file	`dseth.h`
Purpose	To destroy a DSETH socket object.
Parameters	**pSocketObj** Pointer to the DSETH socket object
Return value	Returns the following error codes:

Error Code	Meaning
DSETH_ERR_SUCCESS	No error occurred during the operation. The socket has been deleted.
DSETH_ERR_ILLEGAL	The pSocketObj parameter is illegal (for example, NULL).
DSETH_ERR_COM_DISCONNECTED[1)]	Communication is not possible because the DSETH device is currently not connected. The socket has been deleted nevertheless.
DSETH_ERR_UNINITIALIZED	The DSETH device is physically connected, but is currently not initialized. Communication is not possible. The socket has been deleted nevertheless.
DSETH_ERR_COM_ERROR[1)]	Communication error with the DSETH device. The socket has been deleted nevertheless.
DSETH_ERR_COM_ERROR_TIMEOUT[1)]	Timeout in communication with the DSETH device. The socket has been deleted nevertheless.

[1)] Note: Not relevant for the ETH_TP1 device.

Related topics

Basics

Data Type Definitions.......... 7

Buffer Handling Functions

Introduction

To get information on the functions used for sending and receiving data.

Where to go from here

Information in this section

dsEthDataBufferRXObjFree

Syntax

```
DsEthTError dsEthDataBufferRXObjFree(
     pDsEthSDataBufferObj pDataBufferObj)
```

Include file

```
dseth.h
```

Purpose

To free an RX data buffer object.

Description

Data is received via the `dsEthDataBufferRXObjGet` function. After the received data is used, you must free the RX buffer object using the `dsEthDataBufferRXObjFree` function.

Parameters

pDataBufferObj Pointer to the dsEth data buffer object that was initialized with the `dsEthDataBufferRXObjGet` function.

Return value

Returns the following error codes:

Error Code	Meaning
DSETH_ERR_SUCCESS	No error occurred during the operation.
DSETH_ERR_UNINITIALIZED	Communication is not possible because the device is currently not initialized.
DSETH_ERR_ILLEGAL	Some function parameters are illegal.

Related topics

References

dsEthDataBufferRXObjGet.. 38
dsEthDataBufferTXObjGet.. 39
dsEthDataBufferTXObjSend.. 40

dsEthDataBufferRXObjGet

Syntax

```
pDsEthSDataBufferObj dsEthDataBufferRXObjGet(
          pDsEthSSocketObj pSocketObj)
```

Include file

```
dseth.h
```

Purpose

To receive an RX data buffer object.

Description

Received data is delivered via the `dsEthDataBufferRXObjGet` function. The function delivers the next data received by the related DsEthSSocketObj.

After the received data is used, you must free the RX buffer object. Refer to dsEthDataBufferRXObjFree on page 37.

Parameters

pSocketObj Pointer to the DSETH socket object

Return value

This function returns:

- NULL, if the function failed or new data is available.
- A pointer to a DsEthSDataBufferObj structure.

Related topics

References

dsEthDataBufferTXObjGet

Syntax

```
pDsEthSDataBufferObj dsEthDataBufferTXObjGet(
      pDsEthSSocketObj pSocketObj)
```

Include file

```
dseth.h
```

Purpose

To get a TX data buffer object which is to be sent subsequently.

Description

Data transmission is performed by getting and sending data buffer objects (pDsEthSDataBufferObj).

The `dsEthDataBufferTXObjGet` function gives you a new TX data buffer object. Use the `dsEthDataBufferTXObjSend` function to send the data buffer object extended with the data to be sent.

Parameters

pSocketObj	Pointer to the DSETH socket object

Return value

This function returns:

- NULL, if the function failed.
- A pointer to a new DsEthSDataBufferObj structure.

Related topics

References

dsEthDataBufferTXObjSend

Syntax

```
DsEthTError dsEthDataBufferTXObjSend(
      pDsEthSDataBufferObj pDataBufferObj,
      DSETHUInt32 Flags)
```

Include file

```
dseth.h
```

Purpose

To send a TX data buffer object.

Description

Transmitting data is performed by getting and sending data buffer objects (pDsEthSDataBufferObj).

Use the `dsEthDataBufferTXObjSend` function to send the data buffer object previously obtained via the `dsEthDataBufferTXObjGet` function.

Parameters

pDataBufferObj Pointer to the DSETH data buffer object initialized via the `dsEthDataBufferTXObjGet` function

Flags Masked flags indicating the behavior of the TX functionality. The following flags are available:

Error Code	Meaning
DSETH_USE_BACKGROUND	Data is sent in the background.
DSETH_USE_FOREGROUND	Data is sent in the foreground.
DSETH_WAIT_UNTIL_FINISHED	Waits until the data packet is sent, or the sending attempt timed out until the function returns.
DSETH_FREE_OBJ_AFTER_SEND	Frees the pDataBufferObj object after data is sent. The corresponding data buffer is not available any longer.
DSETH_FREE_OBJ_AFTER_ERROR	Frees the pDataBufferObj object after a data error occurred. Due to the data error, the data was not sent. The corresponding data buffer is not available any longer.

Return value

Returns the following error codes:

Error Code	Meaning
DSETH_ERR_SUCCESS	No error occurred during the operation.
DSETH_ERR_COM_ERROR[1)]	Communication error with the DSETH device.
DSETH_ERR_COM_ERROR_TIMEOUT[1)]	Timeout in communication with the DSETH device.
DSETH_ERR_COM_DISCONNECTED[1)]	Communication is not possible because the DSETH device is currently not connected.
DSETH_ERR_UNINITIALIZED	Communication is not possible because the DSETH device is currently not initialized.
DSETH_ERR_ILLEGAL	Some function parameters are illegal.
DSETH_ERR_FULL	There is no free space in the corresponding FIFO. Data cannot be stored.
DSETH_ERR_DEVICE_NOT_SUPPORTED	The identified device is not supported by the driver.

[1)] Note: Not relevant for the ETH_TP1 device.

Related topics

References

Interrupt Handling Functions

Introduction

To get information on the functions used for interrupt-based data processing.

dsEthInterruptProcess

Syntax

```
DsEthTError dsEthInterruptProcess(pDsEthSObj pDsEthObj)
```

Include file

```
dseth.h
```

Purpose

To perform interrupt-based data processing.

Description

To perform actions when an RX/TX interrupt was detected, the `dsEthInterruptProcess` function must be called from within the interface's interrupt service routine, for example, from within a DS4121 hardware interrupt real-time kernel task.

Parameters

pDsEthObj	Pointer to the DSETH communication object

Return value

Returns the following error codes:

Error Code	Meaning
DSETH_ERR_SUCCESS	No error occurred during the operation.
DSETH_ERR_ILLEGAL	Some function parameters are illegal.

Examples

Where to go from here

Information in this section

Using the DSETH with a DS1006 System and a DS867 LVDS-Ethernet Cable

Example

The following example shows how to use the DSETH interface with a modular system based on DS1006 and a DS867 LVDS Ethernet link cable.

```
#include <Brtenv.h>
#include <ds4121.h>
#include <dseth.h>
#include <phsint.h>
#include <dseth_custom_ds867.h>
/* select the LVDS channel 1 or 2 for DS4121 */
#define CHANNEL_NO 1
#define SOCKET_SEND_CONDITIONS (DSETH_USE_BACKGROUND | DSETH_USE_FOREGROUND | DSETH_FREE_OBJ_AFTER_SEND | \
                                DSETH_FREE_OBJ_AFTER_ERROR)

#define SEND_PERIOD 5.0 /* seconds */
pDsEthSObj pDsEthObj = NULL;
```

```
/* Interrupt service routine */
void DS867InterruptHandler(void)
{
  dsEthInterruptProcess (pDsEthObj);
}
int main(void)
{
  pDsEthCfgSComHandle  pComHandle;
  pDsEthSSocketObj     pSocketObj;

  phs_addr_t board_base;
  /* initialize hardware system  */
  init();
  /* get DS4121 base address of first board */
  board_base = get_peripheral_addr(DS4121_BOARD_ID, 1 /* board number */);
  if (board_base == 0xFFFFFFFF)
  {
    msg_error_printf(MSG_SM_USER, 0, "Error: Board DS4121 was not found!");
    RTLIB_EXIT(1);
  }
  /* initialize DS4121 board */
  ds4121_init(board_base);
  /* install DS4121 service routine for INT0 or INT4 */
  if (CHANNEL_NO == 1)
  {
    install_phs_int_vector(board_base, 0, DS867InterruptHandler);
  }
  else
  {
    install_phs_int_vector(board_base, 4, DS867InterruptHandler);
  }
  /* enable interrupts globally */
  RTLIB_INT_ENABLE();
  /* Creating generic communication handler for DS867 */
  {
    DsEthCstmDS867SComHandle comHandleDS867 = DSETH_CSTM_DS867_COM_HANDLE_INIT_DEFAULT;
    comHandleDS867.Base = board_base;
    comHandleDS867.Channel = CHANNEL_NO; /* LVDS Channel */
    pComHandle = dsEthCstmDS867( &comHandleDS867 );
  }
  /* initialize customization layer */
  dsEthCstmInit(pComHandle);
  /* Creating dsETH object used to access Ethernet interface hardware */
  {
    DsEthSObjInit DsEthObjInit = DSETH_OBJ_INIT_DEFAULT;
    DsEthObjInit.pComHandle = pComHandle;
    pDsEthObj = dsEthObjInit("Eth1", &DsEthObjInit );
  }
  /* creating udp socket */
  {
    static UInt32 LocalSocketID = 0;

    DsEthSSocketInit SocketInit = DSETH_SOCKET_INIT_DEFAULT;

    SocketInit.ConfigFlags = DSETH_ACCESS_MODE_UDP_IP;
    SocketInit.ConfigFlags |= DSETH_ACCESS_SOCKET_DIRECTION_RX;
    SocketInit.ConfigFlags |= DSETH_ACCESS_SOCKET_DIRECTION_TX;

    SocketInit.IPLocal = DSETHIPADDR(192,168,0,1) /* Local: 192.168.0.1:5000 */;
    SocketInit.PortLocal = 5000;
    SocketInit.IPRemote = DSETHIPADDR(192,168,0,2) /* Remote: 192.168.0.2:5000 */;
    SocketInit.PortRemote = 5000;

    SocketInit.pSpecific = &LocalSocketID;

    pSocketObj = dsEthSocketObjInit (pDsEthObj, &SocketInit);
  }
  dsfloat timeStamp = ts_time_read();
  for(;;)
  {
    /* sending data every 5 seconds using udp socket */
    if( (ts_time_read() - timeStamp) >  SEND_PERIOD )
    {
      pDsEthSDataBufferObj pDataBuffer = dsEthDataBufferTXObjGet( pSocketObj );

      UInt8 pData[] = "dSPACE";
      UInt32 Size = 7;

      if( Size > pDataBuffer->SizeMax ) Size = pDataBuffer->SizeMax;

      /* Copying data into data buffer */
      memcpy( pDataBuffer->pData, pData, Size );
      pDataBuffer->Size = Size;

      dsEthDataBufferTXObjSend ( pDataBuffer, SOCKET_SEND_CONDITIONS );

      timeStamp = ts_time_read();
    }
    /* reading all data from udp socket */
    {
      unsigned int LocalSocketID = *((unsigned int *)pSocketObj->pSpecific);
      pDsEthSDataBufferObj pDataBufferObj = NULL;

      pDataBufferObj = dsEthDataBufferRXObjGet( pSocketObj );

      while( NULL != pDataBufferObj)
      {
        msg_info_printf( 0,0," Received Data %s:%d from Socket %d", pDataBufferObj->pData, pDataBufferObj->Size,
                         LocalSocketID );
        dsEthDataBufferRXObjFree( pDataBufferObj );
        pDataBufferObj = dsEthDataBufferRXObjGet( pSocketObj );
      }
    }
    /* calling dsEth and RTLib background service */
    dsEthBackground();
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples

Using the DSETH with a MicroAutoBox II and a DS867 LVDS-Ethernet Cable

Example

The following example shows how to use the DSETH interface with a MicroAutoBox II and a DS867 LVDS Ethernet link cable.

```
#include <Brtenv.h>
#include <dseth.h>
#include <dseth_custom_ds867.h>
#include <Int1401.h>
#define SOCKET_SEND_CONDITIONS (DSETH_USE_BACKGROUND | DSETH_USE_FOREGROUND | DSETH_FREE_OBJ_AFTER_SEND | \
                                DSETH_FREE_OBJ_AFTER_ERROR)

#define SEND_PERIOD 5.0 /* seconds */

pDsEthSObj pDsEthObj = NULL;
/* Interrupt handler function */
void DS867InterruptHandler(void)
{
  dsEthInterruptProcess (pDsEthObj);
}
void main(void)
{
  pDsEthCfgSComHandle  pComHandle;
  pDsEthSSocketObj     pSocketObj;

  /* initialize hardware system  */
  init();
  /* Creating generic communication handler for DS867 */
  {
    DsEthCstmDS867SComHandle comHandleDS867 = DSETH_CSTM_DS867_COM_HANDLE_INIT_DEFAULT;

    comHandleDS867.Version = 0;
    comHandleDS867.Channel = 1;

    comHandleDS867.Base = ECU_TP1_1_MODULE_ADDR; /* 1st ECU_TP1 module */
    pComHandle = dsEthCstmDS867( &comHandleDS867 );
  }
  /* initialize customization layer */
  dsEthCstmInit(pComHandle);
  /* Creating dsETH object used to access Ethernet interface hardware */
  {
    DsEthSObjInit DsEthObjInit = DSETH_OBJ_INIT_DEFAULT;
    DsEthObjInit.pComHandle = pComHandle;
    pDsEthObj = dsEthObjInit("Eth0", &DsEthObjInit );
  }
```

```
/* creating udp socket */
{
  static UInt32 LocalSocketID = 0;

  DsEthSSocketInit SocketInit = DSETH_SOCKET_INIT_DEFAULT;

  SocketInit.ConfigFlags = DSETH_ACCESS_MODE_UDP_IP;
  SocketInit.ConfigFlags |= DSETH_ACCESS_SOCKET_DIRECTION_RX;
  SocketInit.ConfigFlags |= DSETH_ACCESS_SOCKET_DIRECTION_TX;

  SocketInit.IPLocal = DSETHIPADDR(192,168,0,1) /* Local: 192.168.0.1:5000 */;
  SocketInit.PortLocal = 5000;
  SocketInit.IPRemote = DSETHIPADDR(192,168,0,2) /* Remote: 192.168.0.2:5000 */;
  SocketInit.PortRemote = 5000;

  SocketInit.pSpecific = &LocalSocketID;

  pSocketObj = dsEthSocketObjInit (pDsEthObj, &SocketInit);
}
/* Enable ECU interrupt for MicroAutoBox II ECU_TP1 interface */
ds1401_set_interrupt_vector(DS1401_IR14, (DS1401_Int_Handler_Type)DS867InterruptHandler, SAVE_REGS_ON);
ds1401_enable_hardware_int(DS1401_IR14);
/* enable interrupts globally */
RTLIB_INT_ENABLE();
dsfloat timeStamp = ts_time_read();
for(;;)
{
  /* sending data every 5 seconds using udp socket */
  if( (ts_time_read() - timeStamp) >  SEND_PERIOD )
  {
    pDsEthSDataBufferObj pDataBuffer = dsEthDataBufferTXObjGet( pSocketObj );

    UInt8 pData[] = "dSPACE";
    UInt32 Size = 7;

    if( Size > pDataBuffer->SizeMax ) Size = pDataBuffer->SizeMax;

    /* Copying data into data buffer */
    memcpy( pDataBuffer->pData, pData, Size );
    pDataBuffer->Size = Size;

    dsEthDataBufferTXObjSend ( pDataBuffer, SOCKET_SEND_CONDITIONS );

    timeStamp = ts_time_read();
  }
  /* reading all data from udp socket */
  {
    unsigned int LocalSocketID = *((unsigned int *)pSocketObj->pSpecific);
    pDsEthSDataBufferObj pDataBufferObj = NULL;

    pDataBufferObj = dsEthDataBufferRXObjGet( pSocketObj );

    while( NULL != pDataBufferObj)
    {
      msg_info_printf( 0,0," Received Data %s:%d from Socket %d", pDataBufferObj->pData, pDataBufferObj->Size,
                       LocalSocketID );
      dsEthDataBufferRXObjFree( pDataBufferObj );
      pDataBufferObj = dsEthDataBufferRXObjGet( pSocketObj );
    }
  }
```

```
    /* calling dsEth and RTLib background service */
    dsEthBackground();
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples

Using the DSETH with the ETH_TP1 Interface of a MicroAutoBox II

Example

The following example shows how to use the DSETH interface with the ETH_TP1 interface of a MicroAutoBox II.

```
#include <Brtenv.h>
#include <dseth.h>
#include <dseth_custom_eth_tp1.h>
#include <Int1401.h>
#define SOCKET_SEND_CONDITIONS (DSETH_USE_BACKGROUND | DSETH_USE_FOREGROUND | DSETH_FREE_OBJ_AFTER_SEND | \
                                DSETH_FREE_OBJ_AFTER_ERROR)
#define SEND_PERIOD 5.0 /* seconds */
pDsEthSObj pDsEthObj = NULL;
/* Interrupt handler function */
void DS867InterruptHandler(void)
{
  dsEthInterruptProcess (pDsEthObj);
}
void main(void)
{
  pDsEthCfgSComHandle  pComHandle;
  pDsEthSSocketObj     pSocketObj;
  /* initialize hardware system  */
  init();
  /* Creating generic communication handler object for ETH_TP1 interface */
  {
    dsEthCstmEthTp1SComHandle  comHandleEthTp1 = DSETH_CSTM_ETH_TP1_COM_HANDLE_INIT_DEFAULT;
    comHandleEthTp1.ModuleNo = 1; /* Index of ETH_TP1 module */
    pComHandle = dsEthCstmEthTp1( &comHandleEthTp1 );
  }
  /* initialize customization layer */
  dsEthCstmInit(pComHandle);
  /* Creating dsETH object used to access Ethernet interface hardware */
  {
    DsEthSObjInit DsEthObjInit = DSETH_OBJ_INIT_DEFAULT;
    DsEthObjInit.pComHandle = pComHandle;
    pDsEthObj = dsEthObjInit("Eth0", &DsEthObjInit );
  }
  /* creating udp socket */
  {
    static UInt32 LocalSocketID = 0;
    DsEthSSocketInit SocketInit = DSETH_SOCKET_INIT_DEFAULT;
    SocketInit.ConfigFlags = DSETH_ACCESS_MODE_UDP_IP;
    SocketInit.ConfigFlags |= DSETH_ACCESS_SOCKET_DIRECTION_RX;
    SocketInit.ConfigFlags |= DSETH_ACCESS_SOCKET_DIRECTION_TX;
    SocketInit.IPLocal = DSETHIPADDR(192,168,0,1) /* Local: 192.168.0.1:5000 */;
    SocketInit.PortLocal = 5000;
    SocketInit.IPRemote = DSETHIPADDR(192,168,0,2) /* Remote: 192.168.0.2:5000 */;
    SocketInit.PortRemote = 5000;
    SocketInit.pSpecific = &LocalSocketID;
    pSocketObj = dsEthSocketObjInit (pDsEthObj, &SocketInit);
  }
  /* Enable interrupt for MicroAutoBox II ETH_TP1 interface */
  ds1401_set_interrupt_vector(DS1401_INT_BYP_ETH, (DS1401_Int_Handler_Type)DS867InterruptHandler, SAVE_REGS_ON);
  ds1401_enable_hardware_int(DS1401_INT_BYP_ETH);
  /* enable interrupts globally */
  RTLIB_INT_ENABLE();
  dsfloat timeStamp = ts_time_read();
  for(;;)
  {
    /* sending data every 5 seconds using udp socket */
    if( (ts_time_read() - timeStamp) > SEND_PERIOD )
    {
      pDsEthSDataBufferObj pDataBuffer = dsEthDataBufferTXObjGet( pSocketObj );

      UInt8 pData[] = "dSPACE";
      UInt32 Size = 7;

      if( Size > pDataBuffer->SizeMax ) Size = pDataBuffer->SizeMax;

      /* Copying data into data buffer */
      memcpy( pDataBuffer->pData, pData, Size );
      pDataBuffer->Size = Size;
      dsEthDataBufferTXObjSend ( pDataBuffer, SOCKET_SEND_CONDITIONS );
      timeStamp = ts_time_read();
    }
    /* reading all data from udp socket */
    {
      unsigned int LocalSocketID = *((unsigned int *)pSocketObj->pSpecific);
      pDsEthSDataBufferObj pDataBufferObj = NULL;
      pDataBufferObj = dsEthDataBufferRXObjGet( pSocketObj );

      while( NULL != pDataBufferObj)
      {
        msg_info_printf( 0,0," Received Data %s:%d from Socket %d", pDataBufferObj->pData, pDataBufferObj->Size,
                         LocalSocketID );
        dsEthDataBufferRXObjFree( pDataBufferObj );
        pDataBufferObj = dsEthDataBufferRXObjGet( pSocketObj );
      }
    }
    /* calling dsEth and RTLib background service */
    dsEthBackground();
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples