



Protocol interface manual

NetLink-MPI

MPI-Interface

Hilscher Gesellschaft für Systemautomation mbH
Rheinstraße 15
D-65795 Hattersheim
Germany

Tel. +49 (6190) 9907-0
Fax. +49 (6190) 9907-50

Hotline and Support: +49 (6190) 9907-99

Sales email: sales@hilscher.com
Hotline and Support email: hotline@hilscher.com

Web: <http://www.hilscher.com>

Index	Date	Version	Chapter	Revision
1	18.03.02	1.000	all	created

Although this software has been developed with great care and intensively tested, Hilscher Gesellschaft für Systemautomation mbH cannot guarantee the suitability of this software for any purpose not confirmed by us in writing.

Guarantee claims shall be limited to the right to require rectification. Liability for any damages which may have arisen from the use of this software or its documentation shall be limited to cases of intent.

We reserve the right to modify our products and their specifications at any time in as far as this contributes to technical progress. The version of the manual supplied with the software applies.

1 Introduction	4
1.1 Netlink Default Parameter	4
1.2 Netlink Interface	5
2 Message Communication	6
2.1 The MPI-Interface	7
2.1.1 MPI Read and Write Data Block (DB)	8
2.1.2 MPI Read and Write Memory (M)	10
2.1.3 MPI Read and Write IO (I and Q)	12
2.1.4 MPI Read and Write Counter (C)	14
2.1.5 MPI Read and Write Timer (T)	16
2.1.6 MPI Disconnect	18
2.1.7 MPI Get OP Status	20
2.1.8 Error Codes Definitions in MPI Response Messages	22

1 Indroduction

1.1 Netlink Default Parameter

By default the Netlink-MPI is configured on the Ethernet interface to auto-detect between 10/100 Mbaud. The Ethernet RJ 45 connector and its pinning can only be used within a Switch or Hub and not for direct operation at a Notebooks Ethernet card for example. This needs a cross link cable.

The Netlink itself does not have a default IP address to suppress address conflicts when you install Netlink the first time in your Ethernet Network. The IP-Address must be set temporarily at this time via the Hilscher specific NetIdent-Protocol or with a program which supports this functionality. If the Netlink is unconfigured it just reacts on this type of Ethernet broadcast messages. The protocol itself supports multiple Netlinks be active in one Ethernet Segment. After having installed the IP address the Netlink is now reachable for either the IP Device Driver Test program or for the configuration software SyCon. If the Netlink is still unconfigured and you repower it, the NetIdent/SetIp procedure must be redone again.

By default the Netlink has the following MPI-Busparameter be active.

Variable name	Value
Address	0
Baud_rate	187,5 KBaud
Tsl	415 bits
min Tsdr	60 bits
max Tsdr	400 bits
Tqui	1bit
Tset	1bit
Ttr	10000 bits
G	20
HSA	31

These MPI-Parameter can be changed also in accordance to the PROFIBUS communication parameter, so that the Netlink for example can be used for a baudrate up to 12Mbaud.

If the IP address shall be set statically or the MPI-parameter have to be changed in any way, the configuration software SyCon must be used. This tool perfoms a download of these parameter into the Netlinks flash memory.

ATTENTION! If the Netlink is statically configured in its IP address, the NetIdent protocol is enable just only 3 minutes after powering it to prevent misuse during runtime.

The Netlink has two LEDs. The yellow LED reflects the Ethernet-Link status and goes to static on when a Ethernet-link to a switch can be found.

The second duo-LED informs about the status of the firmware. If it is flashing acyclic green the Netlink is not configured in its IP-Address, neither via the Netident Protocol nor via static configuration. If it is static green the Netlink is ready for message communication. During bootup-phase this LEDs is flashing fast red for a short time. This is the time that is needed to load the firmware from external serial FLASH memory to the Netlink program memory. A firmware update is also possible via the SyCon configuration tool. When the firmware is loaded and the Netlink is repowered the firmware update is performed. During this time the LED flashes fast green.

1.2 Netlink Interface

This manual describes the HOST interface of an Netlink-MPI DEVICE. The aim of this manual is to support the integration of these devices into own applications based on the IP device driver functions.

In general only two types of functions need to be called in the Device Driver to send and receive the data to and from the DEVICE. These functions are `DevPutMessage()` and `DevGetMessage()`. `DevPutMessage()` function is used to send a message to the Device while `DevGetMessage()` is used to receive the corresponding response from it. Normally one `DevPutMessage()` command call is followed by one `DevGetMessage()` command call.

2 Message Communication

Via the virtual mailbox interface of the Netlink a protocol-oriented interface is provided. So-called messages are transferred through the send and receive channel.

A message is a common data structure within the HOST transmits acyclic data to or receives data from the DEVICE. The Netlink is a passive DEVICE and will not automatically send a message without being addressed before by a command message.

A message consists of an 8 byte message header, and optional 8 byte telegram header and up to 247 bytes of user data.

- **Message Header** Used by the DEVICE operating system for transporting and addressing the message.
- **Telegram Header** Defines the action for the protocol task. MPI
- **User data** Send or receive data.

The message structure is defined in the RCS_USER.H header file.

	Parameter	Type	Explanation
Message header	msg.rx	byte	Receiver Code
	msg.tx	byte	Transmitter Code
	msg.ln	byte	Data length of the Message
	msg.nr	byte	Identification Code
	msg.a	byte	Response Code
	msg.f	byte	Error Code
	msg.b	byte	Command Code
	msg.e	byte	Extention Code
Extended message header	msg.device_adr	byte	Remote partner address
	msg.data_area	byte	Data area
	msg.data_adr	word	Data address
	msg.data_idx	byte	Data index
	msg.data_cnt	byte	Data quantity
	msg.data_type	byte	Data type
	msg.function	byte	Function code
Telegram User Data	msg.d[0...(x-1)]	byte ... byte	User specific data

General structure of a message

2.1 The MPI-Interface

The MPI-Interface in the Netlink offers the following functions

MPI function overview		Remark
Connect	See remark for Read/Write	The connection is established automatically when using the first read/write command. The DEVICE remembers the initial parameter until the connection is disconnected.
Read/Write	Read and write Data Block (DB)	The connection is established automatically when using the first read/write command. The DEVICE remembers the initial parameter until the connection is disconnected.
	Read and write Memory (M)	
	Read and write IO (I and Q)	
	Read and write Counter (C)	
	Read and write Timer (T)	
Disconnect	MPI disconnect	-
OP Status	MPI Get OP Status	-

The MPI interface can handle one command (read or write) at a time per remote station only. This means that the HOST can activate one command message to the DEVICE and then has to wait for the answer message from the DEVICE before it is allowed to send the next command to this remote station again.

2.1.1 MPI Read and Write Data Block (DB)

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8 9 - 224	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	0x31	command = MPI_Read_Write_DB
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0-255	data area, high byte of offset address in DB
	msg.data_adr	word	0-65534	data address, DB address
	msg.data_idx	byte	0-255	data index, low byte of offset address in DB
	msg.data_cnt	byte	1-216 1-222	data count, number of bytes to be written or to be read
	msg.data_type	byte	5	data type bytestring TASK_TDT_UINT8
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
WRITE_DATA	msg.d[0...(x-1)]	byte array		in write access data to be written

This command allows to read or write Data Registers within S7 components which are supporting the MPI protocol. This can be either done via the MPI interface if any is present or the PROFIBUS interface of the remote station. Normally the MPI interface is driven at 187,5kBaud but should be verified with the corresponding configuration tool before calling the command. Via the standard PROFIBUS line the baudrate is relevant which is configured for the whole PROFIBUS network. Ensure that all active (master) station have the same bus parameter settings, especially the Baudrate, the Highest Station Address and the Target Rotation Time.

If the MPI connection to the remote station is not established and the command is called, the DEVICE automatically establishes the connection in the background before accessing the registers.

The access is divided into 4 parameters. First the remote station address must be specified which is addressed during the access. It's either the MPI address of the remote component or its PROFIBUS address. It depends which physical interface is used for the connection. The value must be fixed in msg.device_adr. A range from 0 to 126 is possible. As next the DB value must be set in msg.data_adr. Allowed values are 0 to 65534. Within the DB there can be an offset specified from where the data is read or where the data is written to. The

offset is splitted into 2 values. The offset is caculated by $\text{offset} = \text{msg.data_area} * 256 + \text{msg.data_idx}$. With that a range of 0 to 65534 is possible. The `msg.data_cnt` is the number of bytes to be read or written. The maximum value here is 222 bytes for read respectively 216 bytes for write. The value in `msg.function` specifies a read or write command.

answer message				
Message header	variable	type	value	signification
	msg.rx	byte	255	receiver = HOST
	msg.tx	byte	3	transmitter = MPI-Task
	msg.ln	byte	9-230 8	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0x31	answer = MPI_Read_Write_DB
	msg.f	byte	f	error, see chapter error definitions in MPI
	msg.b	byte	0	no command
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0-255	data area, high byte of offset adress in DB
	msg.data_adr	word	0-65534	data address, DB address
	msg.data_idx	byte	0-255	data index, low byte of offset address in DB
	msg.data_cnt	byte	1-216 1-222	data count, number of bytes which were written or which are read
	msg.data_type	byte	5	data type bytestring TASK_TDT_UINT8
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
READ_DATA	msg.d[0...(x-1)]	byte array		in read access data which was read

In case of a read command the answer message contains in `msg.d[...]` area the data from the DB. In case of a write command just the extended message is delivered back. If an error happened during the access the variable `msg.f` contains a value unequal 0. The definitions of the error codes can be read in the chapter 'Error code definition in MPI response messages'.

2.1.2 MPI Read and Write Memory (M)

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8 9 - 224	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	0x33	command = MPI_Read_Write_M
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0-65534	data address, Memory address
	msg.data_idx	byte	0	data index, unsued
	msg.data_cnt	byte	1-216 1-222	data count, number of bytes to be written or to be read
	msg.data_type	byte	5	data type bytestring TASK_TDT_UINT8
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
WRITE_DATA	msg.d[0...(x-1)]	byte array		in write access data to be written

This command allows to read or write Merker Registers within S7 components which are supporting the MPI protocol. This can be either done via the MPI interface if any is present or the PROFIBUS interface of the remote station. Normally the MPI interface is driven at 187,5kBaud but should be verified with the corresponding configuration tool before calling the command. Via the standard PROFIBUS line the baudrate is relevant which is configured for the whole PROFIBUS network. Ensure that all active (master) station have the same bus parameter settings, especially the Baudrate, the Highest Station Address and the Target Rotation Time.

If the MPI connection to the remote station is not established and the command is called, the DEVICE automatically establishes the connection in the background before accessing the registers.

The access is divided into 3 parameter. First the remote station address must be specified which is addressed during the access. It's either the MPI address of the remote component or its PROFIBUS address. It depends which physical interface is used for the connection. The value must be fixed in `msg.device_adr`. A range from 0 to 126 is possible. As next the Merkerbyte offset value must be set in `msg.data_adr`. Allowed values are 0 to 65534. The `msg.data_cnt` is the number of bytes to be read or written. The maximum value here is 222 bytes for read respectively 216 bytes for write. The value in `msg.function` specifies a read or write command.

response message				
Message header	variable	type	value	signification
	msg.rx	byte	255	receiver = HOST
	msg.tx	byte	3	transmitter = MPI-Task
	msg.ln	byte	9-230 8	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0x33	response = MPI_Read_Write_M
	msg.f	byte	f	error, see chapter error definitions in MPI
	msg.b	byte	0	no command
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0-65534	data address, Merker address
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	1-216 1-222	data count, number of bytes which were written or which are read
	msg.data_type	byte	5	data type bytestring TASK_TDT_UINT8
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
READ_DATA	msg.d[0...(x-1)]	byte array		in read access data which was read

In case of a read command the answer message contains in `msg.d[...]` area the data from the Merker area. In case of a write command just the extended message header is delivered back. If an error happened during the access the variable `msg.f` contains a value unequal 0. The definitions of the error codes can be read in the chapter 'Error code definition in MPI response messages'.

2.1.3 MPI Read and Write IO (I and Q)

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8 9 - 224	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	0x34	command = MPI_Read_Write_IO
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0 1	data area, input area output area
	msg.data_adr	word	0-65534	data address, IO address
	msg.data_idx	byte	0	data index, unsued
	msg.data_cnt	byte	1-216 1-222	data count, number of bytes to be written or to be read
	msg.data_type	byte	5	data type bytestring TASK_TDT_UINT8
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
WRITE_DATA	msg.d[0...(x-1)]	byte array		in write access data to be written

This command allows to read or write IO Registers within S7 components which are supporting the MPI protocol. This can be either done via the MPI interface if any is present or the PROFIBUS interface of the remote station. Normally the MPI interface is driven at 187,5kBaud but should be verified with the corresponding configuration tool before calling the command. Via the standard PROFIBUS line the baudrate is relevant which is configured for the whole PROFIBUS network. Ensure that all active (master) station have the same bus parameter settings, especially the Baudrate, the Highest Station Address and the Target Rotation Time.

If the MPI connection to the remote station is not established and the command is called, the DEVICE automatically establishes the connection in the background before accessing the registers.

The access is divided into 4 parameters. First the remote station address must be specified which is addressed during the access. It's either the MPI address of the remote component or its PROFIBUS address. It depends which physical interface is used for the connection. The value must be fixed in `msg.device_adr`. A range from 0 to 126 is possible. The parameter `msg.data_area` selects the input area = 0 or the output area = 1. As next the IO-Byte offset address must be set in `msg.data_adr`. Allowed values are 0 to 65534. The `msg.data_cnt` is the number of bytes to be read or written. The maximum value here is 222 bytes for read respectively 216 bytes for write. The value in `msg.function` specifies a read or write command.

response message				
Message header	variable	type	value	signification
	<code>msg.rx</code>	byte	255	receiver = HOST
	<code>msg.tx</code>	byte	3	transmitter = MPI-Task
	<code>msg.ln</code>	byte	9-230 8	length of message read access write access
	<code>msg.nr</code>	byte	j	number of the message
	<code>msg.a</code>	byte	0x34	response = MPI_Read_Write_IO
	<code>msg.f</code>	byte	f	error, see chapter error definitions in MPI
	<code>msg.b</code>	byte	0	no command
	<code>msg.e</code>	byte	0	extension
Extended message header	<code>msg.device_adr</code>	byte	0-126	remote station address
	<code>msg.data_area</code>	byte	0 1	data area, input area output area
	<code>msg.data_adr</code>	word	0-65534	data address, IO address
	<code>msg.data_idx</code>	byte	0	data index, unused
	<code>msg.data_cnt</code>	byte	1-216 1-222	data count, number of bytes which were written or which are read
	<code>msg.data_type</code>	byte	5	data type bytestring TASK_TDT_UINT8
	<code>msg.function</code>	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
READ_DATA	<code>msg.d[0...(x-1)]</code>	byte array		in read access data which was read

In case of a read command the answer message contains in `msg.d[...]` area the data from the I or O area. In case of a write command just the extended message header is delivered back. If an error happened during the access the variable `msg.f` contains a value unequal 0. The definitions of the error codes can be read in the chapter 'Error code definition in MPI response messages'.

2.1.4 MPI Read and Write Counter (C)

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8 10 - 224	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no response
	msg.f	byte	0	no error
	msg.b	byte	0x35	command = MPI_Read_Write_Cnt
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0-65534	data address, counter address
	msg.data_idx	byte	0	data index, unsued
	msg.data_cnt	byte	1-108 1-111	data count, number of counters to be written or to be read
	msg.data_type	byte	6	data type word TASK_TDT_UINT16
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
WRITE_DATA	msg.d[0...(x-1)]	word array		in write access data to be written

This command allows to read or write Counter Registers within S7 components which are supporting the MPI protocol. This can be either done via the MPI interface if any is present or the PROFIBUS interface of the remote station. Normally the MPI interface is driven at 187,5kBaud but should be verified with the corresponding configuration tool before calling the command. Via the standard PROFIBUS line the baudrate is relevant which is configured for the whole PROFIBUS network. Ensure that all active (master) station have the same bus parameter settings, especially the Baudrate, the Highest Station Address and the Target Rotation Time.

If the MPI connection to the remote station is not established and the command is called, the DEVICE automatically establishes the connection in the background before accessing the registers.

The access is divided into 3 parameters. First the remote station address must be specified which is addressed during the access. It's either the MPI address of the remote component or its PROFIBUS address. It depends which physical interface is used for the connection. The value must be fixed in `msg.device_adr`. A range from 0 to 126 is possible. As next the Counter start offset address must be set in `msg.data_adr`. Allowed values are 0 to 65534. The `msg.data_cnt` is the number of counters to be read or written. The maximum value here is 111 counters for read respectively 108 counters for write. The value in `msg.function` specifies a read or write command.

	response message			
Message header	variable	type	value	signification
	msg.rx	byte	255	receiver = HOST
	msg.tx	byte	3	transmitter = MPI-Task
	msg.ln	byte	9-230 8	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0x35	response = MPI_Read_Write_Cnt
	msg.f	byte	f	error, see chapter error definitions in MPI
	msg.b	byte	0	no command
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0-65534	data address, Counter address
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	1-108 1-111	data count, number of counter which were written or which are read
	msg.data_type	byte	6	data type word TASK_TDT_UINT16
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
READ_DATA	msg.d[0...(x-1)]	word array		in read access counter data which was read

In case of a read command the answer message contains in `msg.d[...]` area the data from the counter(s). In case of a write command just the extended message header is delivered back. If an error happened during the access the variable `msg.f` contains a value unequal 0. The definitions of the error codes can be read in the chapter 'Error code definition in MPI response messages'.

2.1.5 MPI Read and Write Timer (T)

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8 10 - 224	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	0x36	command = MPI_Read_Write_Tim
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0-65534	data address, timer address
	msg.data_idx	byte	0	data index, unsued
	msg.data_cnt	byte	1-108 1-111	data count, number of timers to be written or to be read
	msg.data_type	byte	6	data type word TASK_TDT_UINT16
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
WRITE_DATA	msg.d[0...(x-1)]	word array		in write access data to be written

This command allows to read or write Timer Registers within S7 components which are supporting the MPI protocol. This can be either done via the MPI interface if any is present or the PROFIBUS interface of the remote station. Normally the MPI interface is driven at 187,5kBaud but should be verified with the corresponding configuration tool before calling the command. Via the standard PROFIBUS line the baudrate is relevant which is configured for the whole PROFIBUS network. Ensure that all active (master) station have the same bus parameter settings, especially the Baudrate, the Highest Station Address and the Target Rotation Time.

If the MPI connection to the remote station is not established and the command is called, the DEVICE automatically establishes the connection in the background before accessing the registers.

The access is divided into 3 parameters. First the remote station address must be specified which is addressed during the access. It's either the MPI address of the remote component or its PROFIBUS address. It depends which physical interface is used for the connection. The value must be fixed in `msg.device_adr`. A range from 0 to 126 is possible. As next the Timer start offset address must be set in `msg.data_adr`. Allowed values are 0 to 65534. The `msg.data_cnt` is the number of timers to be read or written. The maximum value here is 111 timers for read respectively 108 timers for write. The value in `msg.function` specifies a read or write command.

	response message			
Message header	variable	type	value	signification
	msg.rx	byte	255	receiver = HOST
	msg.tx	byte	3	transmitter = MPI-Task
	msg.ln	byte	9-230 8	length of message read access write access
	msg.nr	byte	j	number of the message
	msg.a	byte	0x36	response = MPI_Read_Write_Tim
	msg.f	byte	f	error, see chapter error definitions in MPI
	msg.b	byte	0	no command
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0-65534	data address, Merker address
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	1-108 1-111	data count, number of timers which were written or which are read
	msg.data_type	byte	6	data type word TASK_TDT_UINT16
	msg.function	byte	1 2	function code TASK_TFC_READ TASK_TFC_WRITE
READ_DATA	msg.d[0...(x-1)]	word array		in read access counter data which was read

In case of a read command the answer message contains in `msg.d[...]` area the data from the timer(s). In case of a write command just the extended message header is delivered back. If an error happened during the access the variable `msg.f` contains a value unequal 0. The definitions of the error codes can be read in the chapter 'Error code definition in MPI response messages'.

2.1.6 MPI Disconnect

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	0x3F	command = MPI_Disconnect
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0	data address, unused
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	0	data count , unused
	msg.data_type	byte	0	data type, unused
	msg.function	byte	0	function code, unused

While the DEVICE automatically establishes the connection to the remote station if not connected previously to it, the disconnect command must be executed by the HOST application.

This service should be executed after having finished the access to the station to close the connection properly and the reserved communication channels (SAPs) are freed again.

Message header	response message			
	variable	type	value	signification
	msg.rx	byte	255	receiver = HOST
	msg.tx	byte	3	transmitter = MPI-Task
	msg.ln	byte	8	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	0x3F	response = MPI_Disconnect
	msg.f	byte	f	error, see chapter error definitions in MPI
	msg.b	byte	0	no command
Extended message header	msg.e	byte	0	extension
	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0	data address, unused
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	0	data count , unused
	msg.data_type	byte	0	data type, unused
	msg.function	byte	0	function code, unused

2.1.7 MPI Get OP Status

command message				
Message header	variable	type	value	signification
	msg.rx	byte	3	receiver = MPI-Task
	msg.tx	byte	255	transmitter = HOST
	msg.ln	byte	8	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	0x32	command = MPI_Get_OP_Status
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0	data address, unused
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	0	data count , unused
	msg.data_type	byte	0	data type, unused
	msg.function	byte	0	function code, unused

This service provides the possibility to read out the current operational status of the remote station. The remote address must be set in `msg.device_adr`. If an error happened during the access the variable `msg.f` contains a value unequal 0. The definitions of the error codes can be read in the chapter 'Error code definition in MPI response messages'.

response message				
Message header	variable	type	value	signification
	msg.rx	byte	255	receiver = HOST
	msg.tx	byte	3	transmitter = MPI-Task
	msg.ln	byte	10	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	0x32	response = MPI_Get_OP_Status
	msg.f	byte	f	error, see chapter error definitions in MPI
	msg.b	byte	0	no command
	msg.e	byte	0	extension
Extended message header	msg.device_adr	byte	0-126	remote station address
	msg.data_area	byte	0	data area, unused
	msg.data_adr	word	0	data address, unused
	msg.data_idx	byte	0	data index, unused
	msg.data_cnt	byte	0	data count , unused
	msg.data_type	byte	0	data type, unused
	msg.function	byte	0	function code, unused
OP_STATUS	msg.d[0-1]	word	0 1 2 3	operational status STOP START RUN UNKNOWN

2.1.8 Error Codes Definitions in MPI Response Messages

error code	signification	error source	help
0 = CON_OK	service could be executed without an error		
1 = CON_UE	timeout from remote station	remote station	remote station has not responded within 1 sec.timeout
2 = CON_RR	resource unavailable	remote station	remote station has no left buffer space for the requested service
3 = CON_RS	requested function of master is not activated within the remote station.	remote station	the connection seems to be closed in the remote station.try to send command again.
17 = CON_NA	no response of the remote station	remote station	check network wiring, check remote address, check baud rate
18 = CON_DS	master not into the logical token ring	network in general	check master DP-Address or highest-station-Addresses of other masters. Examine bus wiring to bus short circuits.
20 = CON_LR	Resource of the local FDL controller not available or not sufficient.	HOST	too many messages. no more segments in DEVICE free
21 = CON_IV	the specified msg.data_cnt parameter invalid	HOST	check the limit of 222 bytes (read) respectively 216 bytes (write) in msg.data_cnt
48 = CON_TO	timeout, the request message was accepted but no indication is sent back by the remote station	remote station	MPI protocol error, or station not present
57 = CON_SE	Sequence fault, internal state machine error. Remote station does not react like awaited or a reconnection was retried while connection is already open or device has no SAPs left to open connection channel	remote station	in case of sequence fault consult support center else retry request service again
0x85 = REJ_IV	specified offset address out of limits or not known in the remote station	HOST	please check msg.data_adr if present or offset parameter in request message
0x86 = REJ_PDU	wrong PDU coding in the MPI response of the remote station	DEVICE	contact hotline
0x87 = REJ_OP	specified length to write or to read results in an access outside the limits	HOST	please check msg.data_cnt length in request message