Protocol API

CC-Link IE Field Basic Slave

V1.2.0

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC180402API02EN | Revision 2 | English | 2019-09 | Released | Public

Table of contents

1 Introduction

1.1 About this document

This manual describes the application interface of the CC-Link IE Field Basic Slave protocol stack.

This manual uses the reduced representation of the packet header in the packet description. The 10 parameters of the packet header are always present in the communication between the application and the stack.

1.2 List of revisions

Rev	Date	Name	Revisions
1	2018-08-31	SBO, HHE	Created
2	2019-09-17	HHE	V1.2.0 has same API as V1.1.0

Table 1: List of Revisions

1.3 Functional overview

The main functionality from application view is:

- configure slave
- exchange of cyclic data
- slave diagnosis

1.4 System requirements

This software package has following system requirements to its environment:

- netX chip as CPU hardware platform

1.5 Intended audience

This manual is suitable for software developers with the following background:

- Knowledge of the programming language C
- Knowledge of the use of the real-time operating system rcX
- Knowledge of the Hilscher Task Layer Reference Model
- Knowledge of the netX DPM Interface
- Knowledge of the IEC 61158 Part 2-6 Type 12 specification documents

1.6 Specifications

The data below applies to the CC-Link IE Field Basic Slave firmware and stack version V1.2.0.

1.6.1 Technical Data

Technical Data

Maximum number of cyclic RY data	128 bytes (1024 bits)
Maximum number of cyclic RX data	128 bytes (1024 bits)
Maximum number of cyclic RWw data	512 words (16 bit)
Maximum number of cyclic RWr data	512 words (16 bit)
Occupied stations	1 … 16 (1 station has 64 bits RY data, 32 words RWw data, 64 bits RX data, and 32 words RWr data)
Acyclic communication	SLMP Server and Client
Data transport layer	Ethernet II, IEEE 802.3, 100 MBit/s

Ports

Cyclic data	61450 (UDP)
Discovery and SLMP Server	61451 (UDP)
SLMP Parameter	45237 (UDP)
SLMP Communication	20000 (UDP)

Firmware/stack available for netX

netX 10	no
netX 50	no
netX 51, netX 52	yes
netX 100, netX 500	yes

1.7 Terms, abbreviations and definitions

Term	Description
AP (-task)	Application (-task) on top of the stack
DPM	Dual Port Memory
HAL	Hardware Abstraction Layer
LFW	Loadable firmware
LOM	Linkable object modules
SHM	Shared memory
SLMP	Seamless messaging protocol
XML	Extended Markup Language
RX	Bit data of the slave station that is updated by cyclic transmission and sent to the master station by the slave station
RY	Bit data of the slave station that is updated by cyclic transmission and sent to the slave station by the master station
RWr	Word data of the slave station that is updated by cyclic transmission and sent to the master station by the slave station
RWw	Word data of the slave station that is updated by cyclic transmission and sent to the slave station by the master station
Transient transmission	Transmission performed upon request

Table 2: Terms, abbreviations and definitions

All variables, parameters and data used in this manual have the LSB/MSB (“Intel”) data format. This corresponds to the convention of the Microsoft C Compiler.

1.8 References to documents

This document refers to the following documents:

[1] Hilscher Gesellschaft für Systemautomation mbH: Dual-Port Memory Interface Manual, netX Dual-Port Memory Interface, Revision 16, DOC060302DPM16EN, English, 2019.

[2] Hilscher Gesellschaft für Systemautomation mbH: Packet API, netX Dual-Port Memory, Packet-based services, Revision 3, DOC161001API03EN, English, 2019.

Table 3: References to documents

2 Getting started / Configuration

Configuration of the slave

The CC-Link IE Field Basic Slave requires configuration parameters

Configuration parameters can be set using the

- configuration software SYCON.net or
- can be set from an application program using a the API.

3 Stack features

3.1 Task structure

The dual-port memory is used for exchange of information, data and packets. Configuration and IO data will be transferred using this way.

The user application only accesses the task located in the highest layer namely the AP task that constitutes the application interface of the CC-Link IE Field Basic Slave stack.

The AP task represents the interface between the CC-Link IE Field Basic Slave protocol stack and the dual-port memory. It is responsible for:

- Control of LEDs
- Diagnosis
- Packet routing
- Update of the I/O data

The triple buffer mechanism provides a consistent synchronous access procedure from both sides (DPM and AP task). The triple buffer technique ensures that the access will always affect the last written cell.

3.2 Diagnosis

The following diagnostic capabilities are provided by the CC-Link IE Field Basic Slave protocol stack: Slave Status.

3.2.1 CC-Link IE Field Basic Slave LED states

LED	Color	State	Meaning
RUN	**LED green:** Indicates the operation status.		
	(green)	On	Station in operation and cyclic transmission in progress.
	(green)	Blinking (2.5 Hz)	Station in operation and cyclic transmission stopped.
	(green)	Flickering (10 Hz)	Station not configured.
	(off)	Off	Station is disconnected.
ERR	**LED red:** Indicates error status.		
	(red)	On	Communication error.
	(red)	Triple Flash	DPM watchdog has expired.
	(off)	Off	Station in normal operation.
L/A Ch0 & Ch1	**LED green**		
	(green)	On	**Link:** The station is linked to the Ethernet, but does not send/receive Ethernet frames.
	(green)	Flickering (load dependent)	**Activity:** The station is linked to the Ethernet and sends/receives Ethernet frames.
	(Off)	Off	The station has no link to the Ethernet.

Table 4: LED states for the CC-Link IE Field Basic Slave protocol

Name	Meaning
RUN	Run
ERR	Error
L/A	Link/Activity

Table 5: LED Names CC-Link IE Field Basic Slave protocol

LED State	Definition
Triple Flash	The indicator shows a sequence of three short flashes (each 200 ms), separated by a short off phase (200 ms). The sequence is finished by a long off phase (1,000 ms).
Blinking (2.5 Hz)	The indicator turns on and off with a frequency of 2.5 Hz: “on” for 200 ms, followed by “off” for 200 ms.
Flickering (10 Hz)	The indicator turns on and off with a frequency of 10 Hz: “on” for 50 ms, followed by “off” for 50 ms.
Flickering (load dependent)	The indicator turns on and off with a frequency of approximately 10 Hz to indicate high Ethernet activity: on for approximately 50 ms, followed by off for 50 ms. The indicator turns on and off in irregular intervals to indicate low Ethernet activity.

Table 6: LED state definitions for the CC-Link IE Field Basic Slave protocol

4 The application interface

4.1 Configuration

4.1.1 Set Configuration service

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	29	Packet Data Length in bytes
ulCmd	UINT32	0xAA00	CCLIEFB_CMD_SET_CONFIG_REQ
Data			
ulSystemFlags	UINT32	0 ... 1 Default: 0	System Flags Bit 0: AUTOSTART(0) / APPLICATION CONTROLLED(1) 0 - communication with a master after a device start is allowed without BUS_ON flag, but the communication will be interrupted if the BUS_ON flag changes state to 0. 1 - communication with master is allowed only with the BUS_ON flag. Bit 2 – 31 reserved.
ulWatchdogTime	UINT32	0, 20 ... 65535 Default: 1000	Host-watchdog time in milliseconds Value 0: watchdog timer deactivated.
ulStationId	UINT32	0.0.0.1 … 223.255.255.254	Station id (similar to IP address, little endian format)
ulSubnetMask	UINT32		Subnet mask (little endian format)
ulDefGatewayId	UINT32	0.0.0.1 … 223.255.255.254 0.0.0.0	Default gateway id (similar to Default Gateway Address, little endian format) Value 0.0.0.0 means no gateway id.
usVendorCode	UINT16		CLPA assigned Vendor Code Hilscher: 0x0352
ulModelCode	UINT32		Assigned by vendor
usEquipmentVersion	UINT16		Assigned by vendor
bNumOccupiedStations	UINT8	1 … 16	Number of occupied stations results into the current amount of data exchanged

Table 7: CCLIEFB_CMD_SET_CONFIG_REQ - Set Configuration request

One station has

- RX data: 64 bits (= 8 bytes)
- RY data: 64 bits (= 8 bytes)
- RWr data: 32 words (= 64 bytes)
- RWw data: 32 words (= 64 bytes)

Sum of RX data, RY data, RWr data, and RWw data

- RX bytes and RY bytes: bNumOccupiedStations * 8 bytes
- RWr words and RWw words: bNumOccupiedStations * 32 words

Packet structure reference

```
typedef struct CCLIEFB_SET_CONFIG_REQ_DATA_Ttag
{
  uint32_t ulSystemFlags;
  uint32_t ulWatchdogTime;

  uint32_t ulStationId;
  uint32_t ulSubnetMask;
  uint32_t ulDefGatewayId;

  uint16_t usVendorCode;
  uint32_t ulModelCode;
  uint16_t usEquipmentVersion;
  uint8_t bNumOccupiedStations;
} CCLIEFB_SET_CONFIG_REQ_DATA_T

typedef struct CCLIEFB_SET_CONFIG_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_SET_CONFIG_REQ_DATA_T tData;
} CCLIEFB_SET_CONFIG_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulSta	UINT32		See section *Status and error* codes on page 28.
ulCmd	UINT32	0xAA01	CCLIEFB_CMD_SET_CONFIG_CNF

Table 8: `CCLIEFB_CMD_SET_CONFIG_CNF` - Set Configuration confirmation

Packet structure reference

```
typedef struct CCLIEFB_SET_CONFIG_CNF_Ttag
{
  HIL_PACKET_HEADER_T tHead;
} CCLIEFB_SET_CONFIG_CNF_T;
```

4.2 Cyclic communication

4.2.1 Data received by slave from network: RWw, RY

The process data layout contains an array of the following structure. The number of elements depends on bNumOccupiedStations of the configuration parameters. The first element starts at offset 0.

```
#define MSK_CCLIEFB_PROCESS_IMAGE_CYCLIC_STATE_DATA_VALID 0x01

#define MSK_CCLIEFB_PROCESS_IMAGE_STATION_STATE_ACTIVE 0x01

typedef struct CCLIEFB_PROCESS_IMAGE_SECTION_RX_Ttag
{
  uint8_t bCyclicState;
  uint8_t bStationState;
  uint16_t usRemoteStationInfo;
  uint16_t usCyclicEndCode;
  uint16_t usReserved;
  uint16_t ausWordData[32];
  uint8_t abBitData[8];
} CCLIEFB_PROCESS_IMAGE_SECTION_RX_T;
```

Variable	Type	Value / range	Description
bCyclicState	UINT8	0 … 1	Bit 0: If set to 1, the entire data is actively transmitted
bStationState	UINT8	0 … 1	Bit 0: If set to 1, the actual station is containing valid data
usRemoteStationInfo	UINT16		Received value from master
usCyclicEndCode	UINT16		Not used
usReserved	UINT16	0	Reserved
ausWordData	UINT16[32]		RWw data (32 words per occupied station)
abBitData	UINT8[8]		RY data (8 byte per occupied station)

Table 9: `CCLIEFB_PROCESS_IMAGE_SECTION_RX_T` – Receive data structure element

4.2.2 Data transmitted by slave to network: RWr, RX

The process data layout contains an array of the following structure. The number of elements depends on bNumOccupiedStations of the configuration parameters. The first element starts at offset 0.

```
#define MSK_CCLIEFB_PROCESS_IMAGE_CYCLIC_STATE_DATA_VALID 0x01

typedef __HIL_PACKED_PRE struct CCLIEFB_PROCESS_IMAGE_SECTION_TX_Ttag
{
  uint8_t bCyclicState;
  uint8_t bCyclicStop; /* master only */
  uint16_t usLocalStationInfo; /* only first block in process image */
  uint16_t ausReserved[2];
  uint16_t ausWordData[32];
  uint8_t abBitData[8];
} __HIL_PACKED_POST CCLIEFB_PROCESS_IMAGE_SECTION_TX_T;
```

Variable	Type	Value / range	Description
bCyclicState	UINT8	0 … 1	Bit 0: If set to 1, the actual station is containing valid data
bCyclicStop	UINT8	0	Unused
usLocalStationInfo	UINT16		Only settable in first array element value transmitted transparently
ausReserved	UINT16[2]	0	Reserved
ausWordData	UINT16[32]		RWr data (32 words per occupied station)
abBitData	UINT8[8]		RX data (8 bytes per occupied station)

Table 10: CCLIEFB_PROCESS_IMAGE_SECTION_TX_T – Transmit data structure element

4.3 Status indications

4.3.1 Registration and deregistration of status indications

4.3.1.1 Register for status indications service

This packet registers an application task for receiving status indications.

If the application does not want to receive those indications anymore, it has to use the service described in *Unregister from status indications service* on page 14.

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0x2F10	HIL_REGISTER_APP_REQ

Table 11: HIL_REGISTER_APP_REQ – Register for status indications request

Packet structure reference

```
typedef struct HIL_REGISTER_APP_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
} HIL_REGISTER_APP_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue handle, unchanged
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0x2F11	HIL_REGISTER_APP_CNF

Table 12: HIL_REGISTER_APP_CNF - Register for status indications confirmation

Packet structure reference

```
typedef struct HIL_REGISTER_APP_CNF_Ttag
{
  HIL_PACKET_HEADER_T   tHead;
} HIL_REGISTER_APP_CNF_T;
```

4.3.1.2 Unregister from status indications service

This packet deregisters an application task from receiving status indications.

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0x2F12	HIL_UNREGISTER_APP_REQ

Table 13: HIL_UNREGISTER_APP_REQ – Unregister from status indications request

Packet structure reference

```
typedef struct HIL_UNREGISTER_APP_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
} HIL_UNREGISTER_APP_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue handle, unchanged
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0x2F13	HIL_UNREGISTER_APP_CNF

Table 14: HIL_UNREGISTER_APP_CNF - Unregister from status indications confirmation

Packet structure reference

```
typedef struct HIL_UNREGISTER_APP_CNF_Ttag
{
  HIL_PACKET_HEADER_T   tHead;
} HIL_UNREGISTER_APP_CNF_T;
```

4.3.2 Status indication

The stack sends this indication to the application on any change.

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	18	Packet Data Length in bytes
ulCmd	UINT32	0xAA20	CCLIEFB_CMD_STATUS_IND
Data			
tOwnStatus. bRunCondition	UINT8		Reserved, do not use
tOwnStatus. bErrorCondition	UINT8		If unequal 0, stack encountered an error
tOwnStatus. bBusOnCondition	UINT8		If unequal 0, stack is set to BusOn
tOwnStatus. bConfiguredCondition	UINT8		If unequal 0, stack is configured
tOwnStatus. ulBusStatus	UINT32		Bit 0: Slave address duplication detected Bit 1: Master arbitration active Bit 2: Other master active Bit 30: This station is slave All other bits are reserved.
tRemoteStatus. abStatusMask	UINT8[8]		Bit 0 in element 0 refers to slave active status All other bits are set to 0.
tLedstatus. bStatusLed	UINT8	0 … 2	Displayed Status LED 0 = Off 1 = On 2 = Blinking **Note:** The status 'Flickering' is not reported with this service.
tLedStatus. bErrorLed	UINT8	0 … 2	Displayed Error LED 0 = Off 1 = On 2 = Blinking **Note:** The status 'Triple Flash' is not reported with this service.

Table 15: `CCLIEFB_CMD_STATUS_IND` – Status indication

Packet structure reference

```
#define VAL_CCLIEFB_STATUS_IND_LED_STATUS_OFF 0
#define VAL_CCLIEFB_STATUS_IND_LED_STATUS_ON 1
#define VAL_CCLIEFB_STATUS_IND_LED_STATUS_BLINKING 2

#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_SLAVE_DUPLICATION 0x00000001
#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_MASTER_ARBITRATION 0x00000002
#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_OTHER_MASTER_ACTIVE 0x00000004
#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_IS_SLAVE 0x40000000

typedef struct CCLIEFB_STATUS_IND_DATA_OWN_STATUS_Ttag
{
  uint8_t bRunCondition;
  uint8_t bErrorCondition;
  uint8_t bBusOnCondition;
  uint8_t bConfiguredCondition;
  uint32_t ulBusStatus;
} CCLIEFB_STATUS_IND_DATA_OWN_STATUS_T;

typedef struct CCLIEFB_STATUS_IND_DATA_REMOTE_STATUS_Ttag
{
  uint8_t abStatusMask[8];
} CCLIEFB_STATUS_IND_DATA_REMOTE_STATUS_T;

typedef struct CCLIEFB_STATUS_IND_DATA_LED_Ttag
{
  uint8_t bStatusLed;
  uint8_t bErrorLed;
} CCLIEFB_STATUS_IND_DATA_LED_T;

typedef struct CCLIEFB_STATUS_IND_DATA_Ttag
{
  CCLIEFB_STATUS_IND_DATA_OWN_STATUS_T tOwnStatus;
  CCLIEFB_STATUS_IND_DATA_REMOTE_STATUS_T tRemoteStatus;
  CCLIEFB_STATUS_IND_DATA_LED_T tLedStatus;
} CCLIEFB_STATUS_IND_DATA_T;

typedef struct CCLIEFB_STATUS_IND_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_STATUS_IND_DATA_T tData;
} CCLIEFB_STATUS_IND_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xAA21	CCLIEFB_CMD_STATUS_RES

Table 16: `CCLIEFB_CMD_STATUS_RES` – Status response

Packet structure reference

```
typedef struct CCLIEFB_STATUS_RES_Ttag
{
  HIL_PACKET_HEADER_T tHead;
} CCLIEFB_STATUS_RES_T;
```

4.4 Get status service

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xAA22	CCLIEFB_CMD_GET_STATUS_REQ

Table 17: `CCLIEFB_CMD_GET_STATUS_REQ` – Get status request

Packet structure reference

```
typedef struct CCLIEFB_CMD_GET_STATUS_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
} CCLIEFB_CMD_GET_STATUS_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	18	Packet Data Length in bytes
ulCmd	UINT32	0xAA23	CCLIEFB_CMD_GET_STATUS_CNF
Data			
tOwnStatus. bRunCondition	UINT8		Reserved, do not use
tOwnStatus. bErrorCondition	UINT8		If unequal 0, stack encountered an error
tOwnStatus. bBusOnCondition	UINT8		If unequal 0, stack is set to BusOn
tOwnStatus. bConfiguredCondition	UINT8		If unequal 0, stack is configured
tOwnStatus. ulBusStatus	UINT32		Bit 0: Slave address duplication detected Bit 1: Master arbitration active Bit 2: Other master active Bit 30: This station is slave All other bits are reserved.
tRemoteStatus. abStatusMask	UINT8[8]		Bit 0 in element 0 refers to slave active status All other bits are set to 0.
tLedstatus. bStatusLed	UINT8	0 … 2	Displayed Status LED 0 = Off 1 = On 2 = Blinking **Note:** The status 'Flickering' is not reported with this service.
tLedStatus. bErrorLed	UINT8	0 … 2	Displayed Error LED 0 = Off 1 = On 2 = Blinking **Note:** The status 'Triple Flash' is not reported with this service.

Table 18: `CCLIEFB_CMD_GET_STATUS_CNF` – Get status confirmation

Packet structure reference

```
#define VAL_CCLIEFB_STATUS_IND_LED_STATUS_OFF 0
#define VAL_CCLIEFB_STATUS_IND_LED_STATUS_ON 1
#define VAL_CCLIEFB_STATUS_IND_LED_STATUS_BLINKING 2

#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_SLAVE_DUPLICATION 0x00000001
#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_MASTER_ARBITRATION 0x00000002
#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_OTHER_MASTER_ACTIVE 0x00000004
#define MSK_CCLIEFB_STATUS_IND_BUS_STATUS_IS_SLAVE 0x40000000

typedef struct CCLIEFB_GET_STATUS_CNF_DATA_OWN_STATUS_Ttag
{
  uint8_t bRunCondition;
  uint8_t bErrorCondition;
  uint8_t bBusOnCondition;
  uint8_t bConfiguredCondition;
  uint32_t ulBusStatus;
} CCLIEFB_GET_STATUS_CNF_DATA_OWN_STATUS_T;

typedef struct CCLIEFB_GET_STATUS_CNF_DATA_REMOTE_STATUS_Ttag
{
  uint8_t abStatusMask[8];
} CCLIEFB_GET_STATUS_CNF_DATA_REMOTE_STATUS_T;

typedef struct CCLIEFB_GET_STATUS_CNF_DATA_LED_Ttag
{
  uint8_t bStatusLed;
  uint8_t bErrorLed;
} CCLIEFB_GET_STATUS_CNF_DATA_LED_T;

typedef struct CCLIEFB_GET_STATUS_CNF_DATA_Ttag
{
  CCLIEFB_GET_STATUS_CNF_DATA_OWN_STATUS_T tOwnStatus;
  CCLIEFB_GET_STATUS_CNF_DATA_REMOTE_STATUS_T tRemoteStatus;
  CCLIEFB_GET_STATUS_CNF_DATA_LED_T tLedStatus;
} CCLIEFB_GET_STATUS_CNF_DATA_T;

typedef struct CCLIEFB_GET_STATUS_CNF_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_GET_STATUS_CNF_DATA_T tData;
} CCLIEFB_GET_STATUS_CNF_T;
```

4.5 Acyclic communication

4.5.1 SLMP Server

The SLMP registration allows the application to add SLMP Request types being handled by the application.

4.5.1.1 Register for receiving SLMP Request indications

The application can use this request several times in order to register different commands and subcommands. The confirmation packet returns the variable `ulSlmpHandle` which differs for each registration of an SLMP command and SLMP subcommand. The application has to store the value of `ulSlmpHandle` in case the application requires to unregister an SLMP command and SLMP subcommand.

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	6	Packet Data Length in bytes
ulCmd	UINT32	0xAA80	`CCLIEFB_CMD_SLMP_REGISTER_INDICATION_REQ`
Data			
usTypeFlags	UINT16		Bit 0: register for SLMP ST type Bit 1: register for SLMP MT type Bit 2: register for SLMP EMT type
usCommand	UINT16		SLMP command
usSubCommand	UINT16		SLMP subcommand

Table 19: `CCLIEFB_CMD_SLMP_REGISTER_INDICATION_REQ` – Register for SLMP Request

Packet structure reference

```
#define MSK_CCLIEFB_SLMP_REGISTER_INDICATION_TYPE_FLAGS_ST 0x0001
#define MSK_CCLIEFB_SLMP_REGISTER_INDICATION_TYPE_FLAGS_MT 0x0002
#define MSK_CCLIEFB_SLMP_REGISTER_INDICATION_TYPE_FLAGS_EMT 0x0004

typedef struct CCLIEFB_SLMP_REGISTER_INDICATION_REQ_DATA_Ttag
{
  uint16_t usTypeFlags;
  uint16_t usCommand;
  uint16_t usSubCommand;
} CCLIEFB_SLMP_REGISTER_INDICATION_REQ_DATA_T;

typedef struct CCLIEFB_SLMP_REGISTER_INDICATION_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_SLMP_REGISTER_INDICATION_REQ_DATA_T tData;
} CCLIEFB_SLMP_REGISTER_INDICATION_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	10	Packet Data Length in bytes
ulSta	UINT32		See section *Status and error* codes on page 28.
ulCmd	UINT32	0xAA81	`CCLIEFB_CMD_SLMP_REGISTER_INDICATION_CNF`
Data			
usTypeFlags	UINT16		Same value as in request
usCommand	UINT16		Same value as in request
usSubCommand	UINT16		Same value as in request
ulSlmpHandle	UINT32		The application has to store the value of `ulSlmpHandle` in case the application requires to unregister an SLMP command and SLMP subcommand.

Table 20: `CCLIEFB_CMD_SLMP_REGISTER_INDICATION_CNF` – Register for SLMP confirmation

Packet structure reference

```
typedef struct CCLIEFB_SLMP_REGISTER_INDICATION_CNF_DATA_Ttag
{
  uint16_t usTypeFlags;
  uint16_t usCommand;
  uint16_t usSubCommand;
  uint32_t ulSlmpHandle;
} CCLIEFB_SLMP_REGISTER_INDICATION_CNF_DATA_T;

typedef struct CCLIEFB_SLMP_REGISTER_INDICATION_CNF_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_SLMP_REGISTER_INDICATION_CNF_DATA_T tData;
} CCLIEFB_SLMP_REGISTER_INDICATION_CNF_T;
```

4.5.1.2 Unregister for receiving SLMP Request indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	4	Packet Data Length in bytes
ulCmd	UINT32	0xAA82	CCLIEFB_CMD_SLMP_UNREGISTER_INDICATION_REQ
Data			
ulSlmpHandle	UINT32		The application has to use the value of ulSlmpHandle received with CCLIEFB_CMD_SLMP_REGISTER_INDICATION_CNF in order to identify an SLMP command and SLMP subcommand.

Table 21: CCLIEFB_CMD_SLMP_UNREGISTER_INDICATION_REQ – Unregister from SLMP Request

Packet structure reference

```
typedef struct CCLIEFB_SLMP_UNREGISTER_INDICATION_REQ_DATA_Ttag
{
  uint32_t ulSlmpHandle;
} CCLIEFB_SLMP_UNREGISTER_INDICATION_REQ_DATA_T;

typedef struct CCLIEFB_SLMP_UNREGISTER_INDICATION_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIES_IF_SLMP_UNREGISTER_INDICATION_REQ_DATA_T tData;
} CCLIES_IF_SLMP_UNREGISTER_INDICATION_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulSta	UINT32		See section *Status and error* codes on page 28.
ulCmd	UINT32	0xAA83	CCLIEFB_CMD_SLMP_UNREGISTER_INDICATION_CNF

Table 22: CCLIEFB_CMD_SLMP_UNREGISTER_INDICATION_CNF – Unregister from SLMP confirmation

Packet structure reference

```
typedef struct CCLIEFB_SLMP_UNREGISTER_INDICATION_CNF_Ttag
{
  HIL_PACKET_HEADER_T                    tHead;
} CCLIEFB_SLMP_UNREGISTER_INDICATION_CNF_T;
```

4.5.1.3 Receive SLMP Request

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	14 + n	Packet Data Length in bytes
ulCmd	UINT32	0xAA84	CCLIEFB_CMD_SLMP_REQUEST_IND
Data			
usReceiveOnPort	UINT16		UDP port on which the request has been received
usFormatType	UINT16	0 … 1	0 = binary (indicates little endian format in abData) 1 = ASCII (indicates big endian format in abData)
usCommand	UINT16		SLMP command
usSubCommand	UINT16		SLMP subcommand
ulTimeoutMs	UINT32		Timeout in milliseconds
usDataBytes	UINT16	n	Number of bytes in abData
abData	UINT8[]		SLMP request data always in binary format

Table 23: CCLIES_IF_CMD_SLMP_REQUEST_IND – Receive SLMP Request indication

Packet structure reference

```
typedef struct CCLIEFB_SLMP_REQUEST_IND_DATA_Ttag
{
  uint16_t usReceivedOnPort;
  uint16_t usFormatType;
  uint16_t usCommand;
  uint16_t usSubCommmand;
  uint32_t ulTimeoutMs;
  uint16_t usDataBytes;
  uint8_t abData[1024];
} CCLIEFB_SLMP_REQUEST_IND_DATA_T;

typedef struct CCLIEFB_SLMP_REQUEST_IND_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_SLMP_REQUEST_IND_DATA_T tData;
} CCLIEFB_SLMP_REQUEST_IND_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	2 + n	Packet Data Length in bytes
ulSta	UINT32		See section *Status and error* codes on page 28.
ulCmd	UINT32	0xAA85	CCLIEFB_CMD_SLMP_REQUEST_RES
Data			
usDataBytes	UINT16	n	Number of bytes in response abData
abData	UINT8[]		

Table 24: CCLIEFB_CMD_SLMP_REQUEST_RES – Receive SLMP Request response

Packet structure reference

```
typedef struct CCLIEFB_SLMP_REQUEST_RES_DATA_Ttag
{
  uint16_t usDataBytes;
  uint8_t abData[1024];
} CCLIEFB_SLMP_REQUEST_RES_DATA_T;

typedef struct CCLIEFB_SLMP_REQUEST_RES_Ttag
{
  TLR_PACKET_HEADER_T                    tHead;
} CCLIEFB_SLMP_REQUEST_RES_T;
```

4.5.2 SLMP Client

4.5.2.1 Send SLMP Request

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	24 + n	Packet Data Length in bytes
ulCmd	UINT32	0xAAAA	CCLIEFB_CMD_SLMP_SEND_REQUEST_REQ
Data			
usProtocol	UINT16	0 … 1	0 = TCP, 1 = UDP
usRequestType	UINT16	0x0050 0x00D0 0x0054 0x00D4 0x0055 0x00D5	0x0050 = ST 0x00D0 = ST push only 0x0054 = MT 0x00D4 = MT push only 0x0055 = EMT 0x00D5 = EMT push only
usFormatType	UINT16	0 … 1	0 = binary (indicates little endian format in abReqData) 1 = ASCII (indicates big endian format in abReqData)
usDstProcNo	UINT16	0 … 0x3FF	
usCommand	UINT16		SLMP command
usSubCommand	UINT16		SLMP subcommand
ulTimeoutMs	UINT32	0x00000001 … 0x00F9FF06 0xFFFFFFFF	Timeout in milliseconds. The stack rounds this value up to the next multiple of 250 ms. 0x00000001 (min. value): 1 ms (rounded up to 250 ms), 0x00F9FF06 (max. value): 65535 *250 ms, value 0xFFFFFFFF: no response required
ulDestAddr	UINT32		Destination station id (similar to IP address, little endian format)
usDestPort	UINT16		Destination port
usTotalReqDataBytes	UINT16		Number of bytes transferred
abReqData	UINT8[]		SLMP request data always in binary format

Table 25: CCLIEFB_CMD_SLMP_SEND_REQUEST_REQ – Send SLMP Request request

Packet structure reference

```
/* usProtocol */
#define CCLIEFB_SLMP_SEND_REQUEST_PROTOCOL_TCP 0
#define CCLIEFB_SLMP_SEND_REQUEST_PROTOCOL_UDP 1

/* usRequestType */
#define CCLIEFB_SLMP_SEND_REQUEST_TYPE_ST 0x0050
#define CCLIEFB_SLMP_SEND_REQUEST_TYPE_ST_PUSHONLY 0x00D0
#define CCLIEFB_SLMP_SEND_REQUEST_TYPE_MT 0x0054
#define CCLIEFB_SLMP_SEND_REQUEST_TYPE_MT_PUSHONLY 0x00D4
#define CCLIEFB_SLMP_SEND_REQUEST_TYPE_EMT 0x0055
#define CCLIEFB_SLMP_SEND_REQUEST_TYPE_EMT_PUSHONLY 0x00D5

/* usFormatType */
#define CCLIEFB_SLMP_SEND_REQUEST_FORMAT_TYPE_BINARY 0
#define CCLIEFB_SLMP_SEND_REQUEST_FORMAT_TYPE_ASCII 1

typedef struct CCLIEFB_SLMP_SEND_REQUEST_REQ_DATA_Ttag
{
  uint16_t usProtocol;
  uint16_t usRequestType;
  uint16_t usFormatType;
  uint16_t usDstProcNo;
  uint16_t usCommand;
  uint16_t usSubCommand;
  uint32_t ulTimeoutMs;
  uint32_t ulDestAddr;
  uint16_t usDestPort;
  uint16_t usTotalReqDataBytes;
  uint8_t abReqData[1024];
} CCLIESFB_SLMP_SEND_REQUEST_REQ_DATA_T;

typedef struct CCLIEFB_SLMP_SEND_REQUEST_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_SLMP_SEND_REQUEST_REQ_DATA_T tData;
} CCLIEFB_SLMP_SEND_REQUEST_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	24	Packet Data Length in bytes
ulSta	UINT32		See section *Status and error* codes on page 28.
ulCmd	UINT32	0xAAAB	CCLIEFB_CMD_SLMP_SEND_REQUEST_CNF
Data			
usFormatType	UINT16	0 … 1	Same value as in request
usDstProcNo	UINT16	0 … 0x3FF	Same value as in request
usCommand	UINT16		Same value as in request
usSubCommand	UINT16		Same value as in request
ulTimeoutMs	UINT32		Same value as in request
ulDestAddr	UINT32		Same value as in request
usDestPort	UINT16		Same value as in request
usTotalResData Bytes	UINT16		Number of bytes in SLMP request response
abResData	UINT8[]		SLMP request confirmation data

Table 26: CCLIEFB_CMD_SLMP_SEND_REQUEST_CNF – Send SLMP Request confirmation

Packet structure reference

```
typedef struct CCLIEFB_SLMP_SEND_REQUEST_CNF_DATA_Ttag
{
  uint16_t usProtocol;
  uint16_t usFrameType;
  uint16_t usFormatType;
  uint16_t usDstProcNo;
  uint16_t usCommand;
  uint16_t usSubCommand;
  uint32_t ulTimeoutMs;
  uint32_t ulDestAddr;
  uint16_t usDestPort;
  uint8_t abResData[1024];
} CCLIEFB_SLMP_SEND_REQUEST_CNF_DATA_T;

typedef struct CCLIEFB_SLMP_SEND_REQUEST_CNF_Ttag
{
  HIL_PACKET_HEADER_T tHead;
  CCLIEFB_SLMP_SEND_REQUEST_CNF_DATA_T tData;
} CCLIEFB_SLMP_SEND_REQUEST_CNF_T;
```

5 Status and error codes

5.1 Common status and error codes

Hexadecimal value	Definition and description
0xC0000001	ERR_HIL_FAIL Common error, detailed error information optionally present in the data area of packet.
0xC0000002	ERR_HIL_UNEXPECTED Unexpected failure.
0xC0000003	ERR_HIL_OUTOFMEMORY Ran out of memory.
0xC0000004	ERR_HIL_UNKNOWN_COMMAND Unknown Command in Packet received.
0xC0000009	ERR_HIL_INVALID_PARAMETER Invalid Parameter in Packet found.
0xC000000C	ERR_HIL_WATCHDOG_TIMEOUT Watchdog error occurred.
0xC000000F	ERR_HIL_PACKET_OUT_OF_SEQ A packet index has been not in the expected sequence.
0xC000001A	ERR_HIL_REQUEST_RUNNING Request is already running.
0xC0000119	ERR_HIL_NOT_CONFIGURED Configuration not available
0xC0000120	ERR_HIL_CONFIGURATION_FAULT General configuration fault.
0xC0000123	ERR_HIL_INSUFFICIENT_LICENSE Insufficient license.
0xC0000124	ERR_HIL_PARAMETER_ERROR Parameter error.
0xC0000127	ERR_HIL_NO_MAC_ADDRESS_AVAILABLE No MAC address available.
0xC0000140	ERR_HIL_NETWORK_FAULT General communication fault.
0xC0000142	ERR_HIL_CONNECTION_TIMEOUT Connection timeout.
0xC0000145	ERR_HIL_CABLE_DISCONNECT Cable disconnected.
0xC0000180	ERR_HIL_BUS_OFF Bus Off flag is set.
0xC0000181	ERR_HIL_CONFIG_LOCK Changing configuration is not allowed.
0xC0000201	ERR_HIL_APPLICATION_ALREADY_REGISTERED Application is already registered.
0xC0000202	ERR_HIL_NO_APPLICATION_REGISTERED No application registered.

Table 27: Common status and error codes

5.2 Status/error codes of the CC-Link IE Field Basic Slave

5.2.1 Status/error codes of the CC-Link IE Field Basic Slave stack

Hexadecimal value	Definition and description
0xC0FE0001	CCLIEFB_ERROR_REQUEST_DESTINATION_PROBLEM
0xC0FE0002	CCLIEFB_ERROR_CONFIGURATION_BUSY
0xC0FE0003	CCLIEFB_ERROR_BUS_IS_ON
0xC0FE0004	CCLIEFB_ERROR_NOT_SUPPORTED
0xC0FE0005	CCLIEFB_ERROR_LWIP_INIT
0xC0FE0006	CCLIEFB_ERROR_INVALID_IP_CONFIG
0xC0FE0007	CCLIEFB_ERROR_INVALID_NUM_OCCUPIED_STATIONS
0xC0FE0008	CCLIEFB_ERROR_INVALID_NUM_STATIONS
0xC0FE0009	CCLIEFB_ERROR_NUM_OCCUPIED_STATIONS_EXCEEDED
0xC0FE000A	CCLIEFB_ERROR_DUPLICATE_STATION
0xC0FE000B	CCLIEFB_ERROR_SLMP_CMD_SUBCMD_ALREADY_REGISTERED
0xC0FE000C	CCLIEFB_ERROR_SLMP_INVALID_REG_HANDLE
0xC0FE000D	CCLIEFB_ERROR_BUS_SCAN_ALREADY_ACTIVE
0xC0FE000E	CCLIEFB_ERROR_BUS_SCAN_NOT_ACTIVE

Table 28: Status/error codes of the CC-Link IE Field Basic Slave stack

5.2.2 Status/error codes of the CC-Link IE Field Basic IF Task

Hexadecimal value	Definition and description
0xC0FF0001	ERR_CCLIEFBIF_NO_MORE_REGISTRATIONS_POSSIBLE No more registrations possible.

Table 29: Status/error codes of the CC-Link IE Field Basic IF Task

5.2.3 Status/error codes of the C-Link IE Field Basic AP Task

Hexadecimal value	Definition Description
0xC1000001	ERR_CCLIEFBAP_COMMAND_INVALID Invalid command received.
0xC1000002	ERR_CCLIEFBAP_INVALID_STARTUP_PARAMETER Invalid Startup Parameter.
0xC1000003	ERR_CCLIEFBAP_NO_MORE_REGISTRATIONS_POSSIBLE No more registrations possible.
0xC1000004	ERR_CCLIEFBAP_DUPLICATE_SLAVE_STATION_ID Duplicate slave station id detected.
0xC1000005	ERR_CCLIEFBAP_OTHER_MASTER_ACTIVE Other master active.
0xC1000006	ERR_CCLIEFBAP_ERROR_DETECTED Error detected.

Table 30: Status/error codes of the CC-Link IE Field Basic AP Task

5.2.4 Status/error codes of the SLMP Endcodes

Hexadecimal value	Definition and description
0xC0F6C059	CCLIEFB_ERROR_SLMP_ENDCODE_CMD_SUBCMD_ERROR
0xC0F6C05C	CCLIEFB_ERROR_SLMP_ENDCODE_ERROR_IN_REQ_MESSAGE
0xC0F6C061	CCLIEFB_ERROR_SLMP_ENDCODE_REQ_LENGTH_DOES_NOT_MATCH
0xC0F6CEE0	CCLIEFB_ERROR_SLMP_ENDCODE_BUSY
0xC0F6CEE1	CCLIEFB_ERROR_SLMP_ENDCODE_REQUEST_SIZE_EXCEEDED_EFFECTIVE_PROCESSING_RANGE
0xC0F6CEE2	CCLIEFB_ERROR_SLMP_ENDCODE_RESPONSE_SIZE_EXCEEDED_EFFECTIVE_PROCESSING_RANGE
0xC0F6CF10	CCLIEFB_ERROR_SLMP_ENDCODE_SPECIFIED_SERVER_INFO_NUMBER_DOES_NOT_EXIST
0xC0F6CF20	CCLIEFB_ERROR_SLMP_ENDCODE_CONTAINED_ITEMS_CANNOT_BE_SET
0xC0F6CF30	CCLIEFB_ERROR_SLMP_ENDCODE_PARAMETER_ID_DOES_NOT_EXIST
0xC0F6CF31	CCLIEFB_ERROR_SLMP_ENDCODE_WRITE_EXCLUSIVE_START_NOT_PERFORMED
0xC0F6CF70	CCLIEFB_ERROR_SLMP_ENDCODE_RELAY_PATH_DESTINATION_COMM_ERROR
0xC0F6CF71	CCLIEFB_ERROR_SLMP_ENDCODE_TIMEOUT_OCCURED

Table 31: Status/error codes of the CC-Link IE Field Basic Slave SLMP Endcodes

6 Appendix

6.1 Accessing the protocol stack by programming the AP task's queue

In general, programming the AP task or the stack has to be performed according to the rules explained in the Hilscher Task Layer Reference Manual. There you can also find more information about the variables discussed in the following.

6.1.1 Getting the receiver task handle of the process queue

To get the handle of the process queue of the CCLIEFB_IF-Task the Macro `TLR_QUE_IDENTIFY()` needs to be used. This macro delivers a pointer to the handle of the intended queue to be accessed (which is returned within the third parameter, `phQue`), if you provide it with the name of the queue (and an instance of your own task). The correct ASCII-queue names for accessing the IF-Task, which you have to use as current value for the first parameter (`pszIdn`), is

ASCII Queue name	Description
`"CCLIEFB_IF_QUE"`	Name of the CC-Link_IE_IF-Task process queue

Table 32: Names of Queues in the CC-Link IE Field Basic Slave Firmware

The returned handle has to be used as value `ulDest` in all initiator packets the AP-Task intends to send to the `CCLIEFB_IF`-Task. This handle is the same handle that has to be used in conjunction with the macros like `TLR_QUE_SENDPACKET_FIFO/LIFO()` for sending a packet to the respective task.

Note: The CCLIEFB_IF-Task provides a common access point to all slave tasks when the AP-Task is not used.

6.2 Legal notes

Copyright

Important notes

Utmost care was/is given in the preparation of the documentation at hand consisting of a user's manual, operating manual and any other document type and accompanying texts. However, errors cannot be ruled out. Therefore, we cannot assume any guarantee or legal responsibility for erroneous information or liability of any kind. You are hereby made aware that descriptions found in the user's manual, the accompanying texts and the documentation neither represent a guarantee nor any indication on proper use as stipulated in the agreement or a promised attribute. It cannot be ruled out that the user's manual, the accompanying texts and the documentation do not completely match the described attributes, standards or any other data for the delivered product. A warranty or guarantee with respect to the correctness or accuracy of the information is not assumed.

We reserve the right to modify our products and the specifications for such as well as the corresponding documentation in the form of a user's manual, operating manual and/or any other document types and accompanying texts at any time and without notice without being required to notify of said modification. Changes shall be taken into account in future manuals and do not represent an obligation of any kind, in particular there shall be no right to have delivered documents revised. The manual delivered with the product shall apply.

Under no circumstances shall Hilscher Gesellschaft für Systemautomation mbH be liable for direct, indirect, ancillary or subsequent damage, or for any loss of income, which may arise after use of the information contained herein.

Liability disclaimer

The hardware and/or software was created and tested by Hilscher Gesellschaft für Systemautomation mbH with utmost care and is made available as is. No warranty can be assumed for the performance or flawlessness of the hardware and/or software under all application conditions and scenarios and the work results achieved by the user when using the hardware and/or software. Liability for any damage that may have occurred as a result of using the hardware and/or software or the corresponding documents shall be limited to an event involving willful intent or a grossly negligent violation of a fundamental contractual obligation. However, the right to assert damages due to a violation of a fundamental contractual obligation shall be limited to contract-typical foreseeable damage.

It is hereby expressly agreed upon in particular that any use or utilization of the hardware and/or software in connection with

- Flight control systems in aviation and aerospace;
- Nuclear fission processes in nuclear power plants;
- Medical devices used for life support and
- Vehicle control systems used in passenger transport

shall be excluded. Use of the hardware and/or software in any of the following areas is strictly prohibited:

- For military purposes or in weaponry;
- For designing, engineering, maintaining or operating nuclear systems;
- In flight safety systems, aviation and flight telecommunications systems;
- In life-support systems;
- In systems in which any malfunction in the hardware and/or software may result in physical injuries or fatalities.

You are hereby made aware that the hardware and/or software was not created for use in hazardous environments, which require fail-safe control mechanisms. Use of the hardware and/or software in this kind of environment shall be at your own risk; any liability for damage or loss due to impermissible use shall be excluded.

Warranty

Hilscher Gesellschaft für Systemautomation mbH hereby guarantees that the software shall run without errors in accordance with the requirements listed in the specifications and that there were no defects on the date of acceptance. The warranty period shall be 12 months commencing as of the date of acceptance or purchase (with express declaration or implied, by customer's conclusive behavior, e.g. putting into operation permanently).

The warranty obligation for equipment (hardware) we produce is 36 months, calculated as of the date of delivery ex works. The aforementioned provisions shall not apply if longer warranty periods are mandatory by law pursuant to Section 438 (1.2) BGB, Section 479 (1) BGB and Section 634a (1) BGB [Bürgerliches Gesetzbuch; German Civil Code] If, despite of all due care taken, the delivered product should have a defect, which already existed at the time of the transfer of risk, it shall be at our discretion to either repair the product or to deliver a replacement product, subject to timely notification of defect.

The warranty obligation shall not apply if the notification of defect is not asserted promptly, if the purchaser or third party has tampered with the products, if the defect is the result of natural wear, was caused by unfavorable operating conditions or is due to violations against our operating regulations or against rules of good electrical engineering practice, or if our request to return the defective object is not promptly complied with.

Costs of support, maintenance, customization and product care

Please be advised that any subsequent improvement shall only be free of charge if a defect is found. Any form of technical support, maintenance and customization is not a warranty service, but instead shall be charged extra.

Additional guarantees

Although the hardware and software was developed and tested in-depth with greatest care, Hilscher Gesellschaft für Systemautomation mbH shall not assume any guarantee for the suitability thereof for any purpose that was not confirmed in writing. No guarantee can be granted whereby the hardware and software satisfies your requirements, or the use of the hardware and/or software is uninterruptable or the hardware and/or software is fault-free.

It cannot be guaranteed that patents and/or ownership privileges have not been infringed upon or violated or that the products are free from third-party influence. No additional guarantees or promises shall be made as to whether the product is market current, free from deficiency in title, or can be integrated or is usable for specific purposes, unless such guarantees or promises are required under existing law and cannot be restricted.

Confidentiality

The customer hereby expressly acknowledges that this document contains trade secrets, information protected by copyright and other patent and ownership privileges as well as any related rights of Hilscher Gesellschaft für Systemautomation mbH. The customer agrees to treat as confidential all of the information made available to customer by Hilscher Gesellschaft für Systemautomation mbH and rights, which were disclosed by Hilscher Gesellschaft für Systemautomation mbH and that were made accessible as well as the terms and conditions of this agreement itself.

The parties hereby agree to one another that the information that each party receives from the other party respectively is and shall remain the intellectual property of said other party, unless provided for otherwise in a contractual agreement.

The customer must not allow any third party to become knowledgeable of this expertise and shall only provide knowledge thereof to authorized users as appropriate and necessary. Companies associated with the customer shall not be deemed third parties. The customer must obligate authorized users to confidentiality. The customer should only use the confidential information in connection with the performances specified in this agreement.

The customer must not use this confidential information to his own advantage or for his own purposes or rather to the advantage or for the purpose of a third party, nor must it be used for commercial purposes and this confidential information must only be used to the extent provided for in this agreement or otherwise to the extent as expressly authorized by the disclosing party in written form. The customer has the right, subject to the obligation to confidentiality, to disclose the terms and conditions of this agreement directly to his legal and financial consultants as would be required for the customer's normal business operation.

Export provisions

The delivered product (including technical data) is subject to the legal export and/or import laws as well as any associated regulations of various countries, especially such laws applicable in Germany and in the United States. The products / hardware / software must not be exported into such countries for which export is prohibited under US American export control laws and its supplementary provisions. You hereby agree to strictly follow the regulations and to yourself be responsible for observing them. You are hereby made aware that you may be required to obtain governmental approval to export, reexport or import the product.

6.3 List of tables

6.4 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
Pune, Delhi, Mumbai
Phone: +91 8888 750 777
E-Mail: info@hilscher.in

Italy
Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Seongnam, Gyeonggi, 463-400
Phone: +82 (0) 31-789-3715
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com