Protocol API

CC-Link IE Field Slave

V1.2.0

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC171005API02EN | Revision 2 | English | 2019-02 | Released | Public

Table of contents

1 Introduction 3
1.1 About this document 3
1.2 List of revisions 3
1.3 Functional overview 3
1.4 System requirements 3
1.5 Intended audience 3
1.6 Specifications 4
1.6.1 Technical Data 4
1.7 Terms, abbreviations and definitions 5
1.8 References to documents 5

2 Getting started / Configuration 6
2.1 Configuration of the slave 6

3 Stack features 7
3.1 Task structure 7
3.2 Diagnosis 7
3.2.1 CC-Link IE Field Slave LED states 8

4 The application interface 9
4.1 Configuration 9
4.1.1 Set Configuration service 9
4.2 Cyclic communication 12
4.2.1 Data received by slave from bus: RWw, RY 12
4.2.2 Data transmitted by slave to bus: RWr, RX 12
4.3 Status indications 13
4.3.1 Registration and deregistration of status indications 13
4.3.1.1 Register for status indications service 13
4.3.1.2 Unregister from status indications service 14
4.3.2 Available indications 15
4.3.2.1 Status indication 15
4.4 Get status service 18
4.5 Detailed node status 22
4.6 Acyclic communication 24
4.6.1 Transient1 24
4.6.1.1 Register for Transient1 received indications 24
4.6.1.2 Unregister from Transient1 received indications 26
4.6.1.3 Receive Transient1 indication 27
4.6.1.4 Send Transient1 frame 28
4.6.2 Transient2 30
4.6.2.1 Register for transient2 received indications 30
4.6.2.2 Unregister from Transient2 received indications 32
4.6.2.3 Receive Transient2 indication 33
4.6.2.4 Send Transient2 frame 35
4.6.3 SLMP 37
4.6.3.1 Register for SLMP received indications 37
4.6.3.2 Unregister from SLMP received indications 39
4.6.3.3 Receive SLMP request indication 40
4.6.3.4 Send SLMP response frame 43
4.6.3.5 Send SLMP error response frame 45

5 Status/Error codes 48
5.1 Error codes: CC-Link IE Field Slave 48
5.2 Status/error codes: CC-Link IE Field SLMP Endcodes 50

6 Appendix 51
6.1 Legal notes 51
6.2 List of tables 55
6.3 Contacts 56

1 Introduction

1.1 About this document

This manual describes the application interface of the CC-Link IE Field 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	2018-02-12	SBO, HHE	Created
2	2019-02-11	HHE, RGÖ	Firmware / stack version V1.2.0
			Section *Set Configuration service*: value range of bStationNumber updated.
			Section *Detailed node status* added.
			Section *Error codes: CC-Link IE Field Slave* updated.
			Section *Status/error codes: CC-Link IE Field SLMP Endcodes* updated.

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 Slave firmware and stack version V1.2.0.

1.6.1 Technical Data

Preliminary Technical Data (subject to change)

Remote Device Station

Maximum number of cyclic RY data	16 bytes (128 bits)
Maximum number of cyclic RX data	16 bytes (128 bits)
Maximum number of cyclic RWw data	64 words (16 bit)
Maximum number of cyclic RWr data	64 words (16 bit)

Intelligent Device Station

Maximum number of cyclic RY data	256 bytes (2048 bits)
Maximum number of cyclic RX data	256 bytes (2048 bits)
Maximum number of cyclic RWw data	1024 words (16 bit)
Maximum number of cyclic RWr data	1024 words (16 bit)

Acyclic communication	SLMP
Data transport layer	Ethernet II, IEEE 802.3, 1GBit/s Full-Duplex

Firmware/stack available for netX

netX 10	no
netX 50, netX 51, netX 52	no
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
Intelligent device station	A node capable of performing 1..n bit data and word data cyclic transmission and transient transmission with the master station, and transient transmission with the slave stations, excluding remote I/O stations. Has client functions or server functions during transient transmission
Remote device station	A node capable of performing 1..n bit data and word data cyclic transmission and transient transmission with the master station, and transient transmission with the slave stations, excluding remote I/O stations. Has server functions during transient transmission
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 14, DOC060302DPM14EN, English, 2018.

[2] Hilscher Gesellschaft für Systemautomation mbH: Packet API, netX Dual-Port Memory, Packet-based services, Revision 1, DOC161001API01EN, English, 2017.

Table 3: References to documents

2 Getting started / Configuration

2.1 Configuration of the slave

The slave can be configured by using different means. This includes the following methods:

- Configuration via SYCON.net
- Set Config packet

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 which constitutes the application interface of the CC-Link IE Field Slave stack.

The AP task represents the interface between the CC-Link IE Field 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 Slave protocol stack:

- Slave Status

3.2.1 CC-Link IE Field Slave LED states

LED	Color	State	Meaning
RUN	**LED green:** Indicates the operation status.		
	(green)	On	Operating normally (depending from the netX firmware “BusOn” status)
	(off)	Off	A watchdog timer error or a hardware failure has occurred.
RD	**LED green:** Displays the reception status of the data.		
	(green)	On	Receiving data.
	(off)	Off	Data not received.
SD	**LED green:** Displays the sending status of the data.		
	(green)	On	Sending data.
	(off)	Off	Data not sent.
D-LINK	**LED yellow:** Indicates the status of the data link.		
	(yellow)	On	Data link in operation (cyclic transmission in progress)
	(yellow)	Blinking	Data link in operation (cyclic transmission stopped)
	(off)	Off	Data link not performed (disconnected)
ERR	**LED red:** Indicates the CP520 error status.		
	(red)	On	Error in own station
	(off)	Off	Normal operation
USER1	**LED yellow:** Indicates an user-defined status 1.		
	(yellow)	On	Currently not used
	(off)	Off	Currently not used
USER2	**LED yellow:** Indicates an user-defined status 2.		
	(yellow)	On	Currently not used
	(off)	Off	Currently not used
LINK0, LINK1 Ch0 & Ch1	**LED green**		
	(green)	On	Link up
	(Off)	Off	Link down
L-ERR0, L-ERR1 Ch0 & Ch1	**LED yellow**		
	(yellow)	On	Abnormal data received or loopback in progress
	(Off)	Off	Normal data received or loopback not performed

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

Name	Meaning
RUN	Run
RD	Reception status of the data
SD	Sending status of the data
D-LINK	Data link

Name	Meaning
ERR	Error
USER	User-defined status
LINK	Link status
L-ERR	Error status of the received data, the line and the loopback

Table 5: LED Names CC-Link IE Field 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		Packet Data Length in bytes
ulCmd	UINT32	0xA700	`CCLIES_IF_CMD_SET_CONFIG_REQ`
ulExt	UINT32	0	Sequence number for use in fragmented packets
Data			
ulSystemFlags	UINT32	0 ... 1 Default: 0	System Flags Bit 0: AUTOSTART(0) / APPLICATION CONTROLLED(1) 0 - communication with a controller 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 controller 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 to be switched off.
bNetworkNumber	UINT8	0, 1 … 239	Network number Unique identifier of the CC-Link Field network 0 = automatic detection by slave 1 … 239 = network number
bStationNumber	UINT8	1 … 254	Station number of the slave Unique identifier of the slave station within the network **Note:** Mitsubishi PLCs use the range 1 … 120.
bDeviceType	UINT8	0x33, 0x34	Device type 0x33 = Intelligent Device Station 0x34 = Remote Device Station
bIOType	UINT8	0 … 3	I/O type 0: mixed (input and output are mixed and the same address is used as the input and output) 1: input 2: output 3: composite (input and output are mixed and the same address is not used as the input and output)

Variable	Type	Value / range	Description
usMaxRWwPoints	UINT16	Intelligent Device: 0 … 1024 Remote Device: 0 … 64	Maximum number of Register Word write (16-bit word)
usMaxRYBytes	UINT16	Intelligent Device: 4 … 256 Remote Device: 4 … 16	Maximum number of Register output bit
usMaxRWrPoints	UINT16	Intelligent Device: 0 … 1024 Remote Device:: 0 … 64	Maximum number of Register Word read (16-bit word)
usMaxRXBytes	UINT16	Intelligent Device:: 0 … 256 Remote Device: 0 … 16	Maximum number of Register input bit
usMinRWwPoints	UINT16	0 … usMaxRWwPoints	Minimum number of Register Word write (16-bit word)
usMinRYBytes	UINT16	4 … usMaxRYBytes	Minimum number of Register output bit
usMinRWrPoints	UINT16	0 … usMaxRWrPoints	Minimum number of Register Word read (16-bit word)
usMinRXBytes	UINT16	0 … usMaxRXBytes	Minimum number of Register input bit
ulConfigFlags	UINT32	0	Reserved
bNetVersion	UINT8		Version information Only used to initialize CC-Link IE communication chip.
usNetModelType	UINT16		Device type assigned by CLPA
ulNetUnitModelCode	UINT32		Vendor assigned product code
usNetVendorCode	UINT16		Vendor code assigned by CLPA
abNetUnitModelName	UINT8[20]		Vendor assigned product code as string
abNetVendorName	UINT8[32]		Vendor name Vendor code as string
usHwVersion	UINT16		Hardware version Only used to initialize CC-Link IE communication chip.
usDeviceVersion	UINT16		Device version Only used to initialize CC-Link IE communication chip.
bInformationFlag	UINT8	0	Reserved
bCtrlVersion	UINT8	0	Reserved
usCtrlModelType	UINT16	0	Reserved
ulCtrlUnitModelCode	UINT32	0	Reserved
usCtrlVendorCode	UINT16	0	Reserved
abCtrlUnitModelName	UINT8[20]	0	Reserved
abCtrlVendorName	UINT8[32]	0	Reserved
ulVendorInformation	UINT32	0	Reserved
usUnitVersion	UINT16	0	Reserved
abIpNetworkSegment	UINT8[2]	0	Reserved
bDefGatewayNode	UINT8	0	Reserved
bReserved	UINT8	0	Set to zero
aulReserved	UINT32[5]	0	Set to zero

Table 6: CCLIES_IF_CMD_SET_CONFIG_REQ - Set Configuration request

Packet structure reference

```
typedef struct CCLIES_IF_SET_CONFIG_REQ_DATA_Ttag
{
  uint32_t ulSystemFlags;
  uint32_t ulWatchdogTime;
  uint8_t bNetworkNumber;
  uint8_t bStationNumber;
  uint8_t bDeviceType;
  uint8_t bIOType;
  uint16_t usMaxRWwPoints;
  uint16_t usMaxRYBytes;
  uint16_t usMaxRWrPoints;
  uint16_t usMaxRXBytes;
  uint16_t usMinRWwPoints;
  uint16_t usMinRYBytes;
  uint16_t usMinRWrPoints;
  uint16_t usMinRXBytes;
  uint32_t ulConfigFlags;
  uint8_t bNetVersion;
  uint16_t usNetModelType;
  uint32_t ulNetUnitModelCode;
  uint16_t usNetVendorCode;
  uint8_t abNetUnitModelName[20];
  uint8_t abNetVendorName[32];
  uint16_t usHwVersion;
  uint16_t usDeviceVersion;
  uint8_t bInformationFlag;
  uint8_t bCtrlVersion;
  uint16_t usCtrlModelType;
  uint32_t ulCtrlUnitModelCode;
  uint16_t usCtrlVendorCode;
  uint8_t abCtrlUnitModelName[20];
  uint8_t abCtrlVendorName[32];
  uint32_t ulVendorInformation;
  uint16_t usUnitVersion;
  uint8_t abIpNetworkSegment[2];
  uint8_t bDefGatewayNode;
  uint8_t bReserved;
  uint32_t aulReserved[5];
} CCLIES_IF_SET_CONFIG_REQ_DATA_T;

typedef struct CCLIES_IF_SET_CONFIG_REQ_Ttag
{
  TLR_PACKET_HEADER_T                      tHead;
  CCLIES_IF_SET_CONFIG_REQ_DATA_T tData;
} CCLIES_IF_SET_CONFIG_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA701	CCLIES_IF_CMD_SET_CONFIG_CNF

Table 7: CCLIES_IF_CMD_SET_CONFIG_CNF - Set Configuration confirmation

Packet structure reference

```
typedef struct CCLIES_IF_SET_CONFIG_CNF_Ttag
{
  TLR_PACKET_HEADER_T                      tHead;
} CCLIES_IF_SET_CONFIG_CNF_T;
```

4.2 Cyclic communication

4.2.1 Data received by slave from bus: RWw, RY

RWw words start at byte offset 0 and have the maximum length of usMaxRWwPoints 16 bit words (usMaxRWwPoints * 2 bytes).

RY words start at byte offset (usMaxRWwPoints * 2) and have the maximum length of usMaxRYBytes.

Note: The offset address of RWw, RY words is 4-byte aligned.

4.2.2 Data transmitted by slave to bus: RWr, RX

RWr words start at byte offset 0 and have the maximum length of usMaxRWrPoints 16 bit words (usMaxRWrPoints * 2 bytes).

RX words start at byte offset (usMaxRWrPoints * 2) and have the maximum length of usMaxRXBytes.

Note: The offset address of RWr, RX words is 4-byte aligned.

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.

The following groups of status indications will be sent to the application task after successful registration: see section *Available indications* on page 15.

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	RCX_REGISTER_APP_REQ

Table 8: RCX_REGISTER_APP_REQ – Register for status indications request

Packet structure reference

```
typedef struct RCX_REGISTER_APP_REQ_Ttag
{
  TLR_PACKET_HEADER_T tHead;
} RCX_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	RCX_REGISTER_APP_CNF

Table 9: RCX_REGISTER_APP_CNF - Register for status indications confirmation

Packet structure reference

```
typedef struct RCX_REGISTER_APP_CNF_Ttag
{
  TLR_PACKET_HEADER_T   tHead;
} RCX_REGISTER_APP_CNF_T;
```

4.3.1.2 Unregister from status indications service

This packet deregisters an application task from receiving status indications.

The following status indications will not continue to be sent to the application task anymore after successful deregistration: see section *Available indications* on page 15.

Packet description

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

Table 10: RCX_UNREGISTER_APP_REQ – Unregister from status indications request

Packet structure reference

```
typedef struct RCX_UNREGISTER_APP_REQ_Ttag
{
  TLR_PACKET_HEADER_T tHead;
} RCX_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	RCX_UNREGISTER_APP_CNF

Table 11: RCX_UNREGISTER_APP_CNF - Unregister from status indications confirmation

Packet structure reference

```
typedef struct RCX_UNREGISTER_APP_CNF_Ttag
{
  TLR_PACKET_HEADER_T   tHead;
} RCX_UNREGISTER_APP_CNF_T;
```

4.3.2 Available indications

4.3.2.1 Status indication

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	22	Packet Data Length in bytes
ulCmd	UINT32	0xA720	`CCLIES_IF_CMD_STATUS_IND`
Data			
tLedStatus. bRunLed	UINT8	0, 1, 2	Status of RUN led
tLedStatus. bErrorLed	UINT8	0, 1	Status of ERR led
tLedStatus. bDlinkLed	UINT8	0, 1, 2	Status of D-Link led
tLedStatus. bLErr1Led	UINT8	0, 1	Status of LErr1 led
tLedStatus. bLErr2Led	UINT8	0, 1	Status of LErr2 led
tLedStatus. bSdLed	UINT8	0, 1	Status of SD led
tLedStatus. bRdLed	UINT8	0, 1	Status of RD led
tMasterStatus. bRunCondition	UINT8	0, 1, 2, 3	Detailed Application Operation Status, see *Table 13* below
tMasterStatus. bErrorCondition	UINT8	0, 1, 2, 3	Error Detection Status, see *Table 14* below
tMasterStatus. fLostMaster	UINT8		
tMasterStatus. usCyclicStatus	UINT16		
tMasterStatus. fMasterAppRun	UINT8		
tMasterStatus. fMasterAppErr	UINT8		
tProcessData. usCurrentRWwPoints	UINT16		
tProcessData. usCurrentRYBytes	UINT16		
tProcessData. usCurrentRWrPoints	UINT16		
tProcessData. usCurrentRXBytes	UINT16		

Table 12: `CCLIES_IF_CMD_STATUS_IND` – Status indication

Packet structure reference

```
typedef struct CCLIES_IF_STATUS_IND_DATA_LED_Ttag
{
  uint8_t bRunLed;
  uint8_t bErrorLed;
  uint8_t bDlinkLed;
  uint8_t bLErr1Led;
  uint8_t bLErr2Led;
  uint8_t bSdLed;
  uint8_t bRdLed;
} CCLIES_IF_STATUS_IND_DATA_LED_T;

typedef struct CCLIES_IF_STATUS_IND_DATA_MASTER_Ttag
{
  uint8_t bRunCondition;
  uint8_t bErrorCondition;
  uint8_t fLostMaster;
  uint16_t usCyclicStatus;
  uint8_t fMasterAppRun;
  uint8_t fMasterAppErr;
} CCLIES_IF_STATUS_IND_DATA_MASTER_T;

typedef struct CCLIES_IF_STATUS_IND_DATA_PROCESSDATA_Ttag
{
  uint16_t usCurrentRWwPoints;
  uint16_t usCurrentRYBytes;
  uint16_t usCurrentRWrPoints;
  uint16_t usCurrentRXBytes;
} CCLIES_IF_STATUS_IND_DATA_PROCESSDATA_T;

typedef struct CCLIES_IF_STATUS_IND_DATA_Ttag
{
  CCLIES_IF_STATUS_IND_DATA_LED_T tLedStatus;
  CCLIES_IF_STATUS_IND_DATA_MASTER_T tMasterStatus;
  CCLIES_IF_STATUS_IND_DATA_PROCESSDATA_T tProcessData;
} CCLIES_IF_STATUS_IND_DATA_T;

typedef struct CCLIES_IF_STATUS_IND_Ttag
{
  TLR_PACKET_HEADER_T         tHead;
  CCLIES_IF_STATUS_IND_DATA_T tData;
} CCLIES_IF_STATUS_IND_T;
```

The meaning of the Detailed Application Operation Status (`tMasterStatus.bRunCondition`) is defined as follows:

Value	Meaning
0	Detailed application operation status notification not supported
1	Application stopped
2	Application running
3	Application user does not exist

Table 13: Detailed Application Operation Status

The meaning of the Error Detection Status (`tMasterStatus.bErrorCondition`) is defined as follows:

Value	Meaning
0	No error
1	Minor error
2	Major error
3	Severe error

Table 14: Error Detection Status

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA721	`CCLIES_IF_CMD_STATUS_RES`

Table 15: `CCLIES_IF_CMD_STATUS_RES` – Status response

Packet structure reference

```
typedef struct CCLIES_IF_STATUS_RES_Ttag
{
  TLR_PACKET_HEADER_T                    tHead;
} CCLIES_IF_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	0xA722	CCLIES_IF_CMD_GET_STATUS_REQ

Table 16: CCLIES_IF_CMD_GET_STATUS_REQ – Get status request

Packet structure reference

```
typedef struct CCLIES_IF_GET_STATUS_REQ_Ttag
{
  TLR_PACKET_HEADER_T                       tHead;
} CCLIES_IF_GET_STATUS_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	4	Packet Data Length in bytes
ulCmd	UINT32	0xA723	CCLIES_IF_CMD_GET_STATUS_CNF
Data			
tLedStatus. bRunLed	UINT8	0, 1, 2	Status of RUN led
tLedStatus. bErrorLed	UINT8	0, 1	Status of ERR led
tLedStatus. bDlinkLed	UINT8	0, 1, 2	Status of D-Link led
tLedStatus. bLErr1Led	UINT8	0, 1	Status of LErr1 led
tLedStatus. bLErr2Led	UINT8	0, 1	Status of LErr2 led
tLedStatus. bSdLed	UINT8	0, 1	Status of SD led
tLedStatus. bRdLed	UINT8	0, 1	Status of RD led
tMasterStatus. bRunCondition	UINT8	0, 1, 2, 3	Detailed Application Operation Status, see *Table 18* below
tMasterStatus. bErrorCondition	UINT8	0, 1, 2, 3	Error Detection Status, see *Table 19* below
tMasterStatus. fLostMaster	UINT8		
tMasterStatus. usCyclicStatus	UINT16		
tMasterStatus. fMasterAppRun	UINT8		
tMasterStatus. fMasterAppErr	UINT8		
tProcessData. usCurrentRWwPoints	UINT16		
tProcessData. usCurrentRYBytes	UINT16		
tProcessData. usCurrentRWrPoints	UINT16		
tProcessData. usCurrentRXBytes	UINT16		

Table 17: CCLIES_IF_CMD_GET_STATUS_CNF – Get status confirmation

Packet structure reference

```
typedef struct CCLIES_IF_GET_STATUS_CNF_DATA_LED_Ttag
{
  uint8_t bRunLed;
  uint8_t bErrorLed;
  uint8_t bDlinkLed;
  uint8_t bLErr1Led;
  uint8_t bLErr2Led;
  uint8_t bSdLed;
  uint8_t bRdLed;
} CCLIES_IF_GET_STATUS_CNF_DATA_LED_T;

typedef struct CCLIES_IF_GET_STATUS_CNF_DATA_MASTER_Ttag
{
  uint8_t bRunCondition;
  uint8_t bErrorCondition;
  uint8_t fLostMaster;
  uint16_t usCyclicStatus;
  uint8_t fMasterAppRun;
  uint8_t fMasterAppErr;
} CCLIES_IF_GET_STATUS_CNF_DATA_MASTER_T;

typedef struct CCLIES_IF_GET_STATUS_CNF_DATA_PROCESSDATA_Ttag
{
  uint16_t usCurrentRWwPoints;
  uint16_t usCurrentRYBytes;
  uint16_t usCurrentRWrPoints;
  uint16_t usCurrentRXBytes;
} CCLIES_IF_GET_STATUS_CNF_DATA_RPOCESSDATA_T;

typedef struct CCLIES_IF_GET_STATUS_CNF_DATA_Ttag
{
  CCLIES_IF_GET_STATUS_CNF_DATA_LED_T tLedStatus;
  CCLIES_IF_GET_STATUS_CNF_DATA_MASTER_T tMasterStatus;
  CCLIES_IF_GET_STATUS_CNF_DATA_PROCESSDATA_T tProcessData;
} CCLIES_IF_GET_STATUS_CNF_DATA_T;

typedef struct CCLIES_IF_GET_STATUS_CNF_Ttag
{
  TLR_PACKET_HEADER_T             tHead;
  CCLIES_IF_GET_STATUS_CNF_DATA_T tData;
} CCLIES_IF_GET_STATUS_CNF_T;
```

The meaning of the Detailed Application Operation Status (`tMasterStatus.bRunCondition`) is defined as follows:

Value	Meaning
0	Detailed application operation status notification not supported
1	Application stopped
2	Application running
3	Application user does not exist

Table 18: Detailed Application Operation Status

The meaning of the Error Detection Status (`tMasterStatus.bErrorCondition`) is defined as follows:

Value	Meaning
0	No error
1	Minor error
2	Major error
3	Severe error

Table 19: Error Detection Status

4.5 Detailed node status

Using this packet, the application can set the detailed node status.

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	2	Packet Data Length in bytes
ulCmd	UINT32	0xA724	CCLIES_IF_CMD_SET_DET_NODE_STATUS_REQ
Data			
bDetailedRunStatus	UINT8	0…2	Detailed run status 0: Detailed run status notification not supported 1: Application stopped or pause 2: Application running or executed in single steps
bDetailedErrorStatus	UINT8	0…3	Detailed error status 0: No error 1: Minor error 2: Major error 3: Critical error

Table 20: CCLIES_IF_CMD_SET_DET_NODE_STATUS_REQ – Set detailed node status request

Packet structure reference

```
/* bDetailedRunStatus */
#define CCLIES_IF_SET_DET_NODE_STATUS_NONE 0
#define CCLIES_IF_SET_DET_NODE_STATUS_STOP_OR_PAUSE 1
#define CCLIES_IF_SET_DET_NODE_STATUS_RUN_OR_STEP 2

/* bDetailedErrorStatus */
#define CCLIES_IF_SET_DET_NODE_ERROR_NONE 0
#define CCLIES_IF_SET_DET_NODE_ERROR_LIGHT 1
#define CCLIES_IF_SET_DET_NODE_ERROR_MEDIUM 2
#define CCLIES_IF_SET_DET_NODE_ERROR_CRITICAL 3

/* request packet */
typedef  struct CCLIES_IF_SET_DET_NODE_STATUS_REQ_DATA_Ttag
{
  uint8_t bDetailedRunStatus;
  uint8_t bDetailedErrorStatus;
}   CCLIES_IF_SET_DET_NODE_STATUS_REQ_DATA_T;

typedef struct CCLIES_IF_SET_DET_NODE_STATUS_REQ_Ttag
{
  HIL_PACKET_HEADER_T tHead;
    CCLIES_IF_SET_DET_NODE_STATUS_REQ_DATA_T tData;
}   CCLIES_IF_SET_DET_NODE_STATUS_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA725	CCLIES_IF_CMD_SET_DET_NODE_STATUS_CNF

Table 21: CCLIES_IF_CMD_SET_DET_NODE_STATUS_CNF – Set detected node status confirmation

Packet structure reference

```
/* confirmation packet */
typedef struct CCLIES_IF_SET_DET_NODE_STATUS_CNF_Ttag
{
  HIL_PACKET_HEADER_T tHead;
} CCLIES_IF_SET_DET_NODE_STATUS_CNF_T;
```

4.6 Acyclic communication

4.6.1 Transient1

The Transient1 registration allows adding protocols not being addressed by CC-Link IE Field Slave stack.

The following protocols cannot be registered and are always handled by stack.

Datatype	DataSubtype	Meaning
0x05	0x0002	CC-Link IE Field Common - SLMP
0x07	0x0002	CC-Link IE Field System

Table 22: CC-Link IE Field pre-registered Transient1 protocols

4.6.1.1 Register for Transient1 received indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	3	Packet Data Length in bytes
ulCmd	UINT32	0xA740	`CCLIES_IF_CMD_REGISTER_TRANSIENT1_REQ`
Data			
bDatatype	UINT8		
usDataSubtype	UINT16		

Table 23: `CCLIES_IF_CMD_REGISTER_TRANSIENT1_REQ` - Register for Transient1 request

Packet structure reference

```
typedef struct CCLIES_IF_REGISTER_TRANSIENT1_REQ_DATA_Ttag
{
  uint8_t bDatatype;
  uint16_t usDataSubtype;
} CCLIES_IF_REGISTER_TRANSIENT1_REQ_DATA_T;

typedef struct CCLIES_IF_REGISTER_TRANSIENT1_REQ_Ttag
{
  TLR_PACKET_HEADER_T                      tHead;
  CCLIES_IF_REGISTER_TRANSIENT1_REQ_DATA_T tData;
} CCLIES_IF_REGISTER_TRANSIENT1_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	7	Packet Data Length in bytes
ulCmd	UINT32	0xA741	CCLIES_IF_CMD_REGISTER_TRANSIENT1_CNF
Data			
bDatatype	UINT8		Same value as in request
usDataSubtype	UINT16		Same value as in request
ulTransient1Handle	UINT32		

Table 24: CCLIES_IF_CMD_REGISTER_TRANSIENT1_CNF – Register for Transient1 confirmation

Packet structure reference

```
typedef struct CCLIES_IF_REGISTER_TRANSIENT1_CNF_DATA_Ttag
{
  uint8_t bDatatype;
  uint16_t usDataSubtype;
  uint32_t ulTransient1Handle;
} CCLIES_IF_REGISTER_TRANSIENT1_CNF_DATA_T;

typedef struct CCLIES_IF_REGISTER_TRANSIENT1_CNF_Ttag
{
  TLR_PACKET_HEADER_T                       tHead;
  CCLIES_IF_REGISTER_TRANSIENT1_CNF_DATA_T tData;
} CCLIES_IF_REGISTER_TRANSIENT1_CNF_T;
```

4.6.1.2 Unregister from Transient1 received indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	4	Packet Data Length in bytes
ulCmd	UINT32	0xA742	CCLIES_IF_CMD_UNREGISTER_TRANSIENT1_REQ
Data			
ulTransient1Handle	UINT32		

Table 25: CCLIES_IF_CMD_UNREGISTER_TRANSIENT1_REQ – Unregister from Transient1 request

Packet structure reference

```
typedef struct CCLIES_IF_UNREGISTER_TRANSIENT1_REQ_DATA_Ttag
{
  uint32_t ulTransient1Handle;
} CCLIES_IF_UNREGISTER_TRANSIENT1_REQ_DATA_T;

typedef struct CCLIES_IF_UNREGISTER_TRANSIENT1_REQ_Ttag
{
  TLR_PACKET_HEADER_T                          tHead;
  CCLIES_IF_UNREGISTER_TRANSIENT1_REQ_DATA_T tData;
} CCLIES_IF_UNREGISTER_TRANSIENT1_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA743	CCLIES_IF_CMD_UNREGISTER_TRANSIENT1_CNF

Table 26: CCLIES_IF_CMD_UNREGISTER_TRANSIENT1_CNF – Unregister from Transient1 confirmation

Packet structure reference

```
typedef struct CCLIES_IF_UNREGISTER_TRANSIENT1_CNF_Ttag
{
  TLR_PACKET_HEADER_T                       tHead;
} CCLIES_IF_UNREGISTER_TRANSIENT1_CNF_T;
```

4.6.1.3 Receive Transient1 indication

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	9 + n	Packet Data Length in bytes
ulCmd	UINT32	0xA746	CCLIES_IF_CMD_RECV_TRANSIENT1_IND
Data			
usNetworkID	UINT16		Network number
usNodeID	UINT16		Node number
bDataClassification	UINT8		Transient1 data classification
usDataSubClassification	UINT16		Transient1 data subclassification
usTransientDataAllSize	UINT16		Size of Transient1 data
abData	UINT8[]		Data of transient frame

Table 27: CCLIES_IF_CMD_RECV_TRANSIENT1_IND – Receive Transient1 indication

Packet structure reference

```
typedef struct CCLIES_IF_TRANSIENT1_DATA_Ttag
{
  uint16_t usNetworkID;
  uint16_t usNodeID;
  uint8_t bDataClassification;
  uint16_t usDataSubClassification;
  uint16_t usTransientDataAllSize;
  uint8_t abData[1024];
} CCLIES_IF_TRANSIENT1_DATA_T;

typedef struct CCLIES_IF_RECV_TRANSIENT1_IND_Ttag
{
  TLR_PACKET_HEADER_T         tHead;
  CCLIES_IF_TRANSIENT1_DATA_T tData;
} CCLIES_IF_RECV_TRANSIENT1_IND_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA747	CCLIES_IF_CMD_RECV_TRANSIENT1_RES

Table 28: CCLIES_IF_CMD_RECV_TRANSIENT1_RES – Receive Transient1 response

Packet structure reference

```
typedef struct CCLIES_IF_RECV_TRANSIENT1_RES_Ttag
{
  TLR_PACKET_HEADER_T                     tHead;
} CCLIES_IF_RECV_TRANSIENT1_RES_T;
```

4.6.1.4 Send Transient1 frame

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	9 + n	Packet Data Length in bytes
ulCmd	UINT32	0xA744	CCLIES_IF_CMD_SEND_TRANSIENT1_REQ
Data			
usNetworkID	UINT16		Network number
usNodeID	UINT16		Node number
bDataClassification	UINT8		Transient1 data classification
usDataSubClassification	UINT16		Transient1 data subclassification
usTransientDataAllSize	UINT16		Size of Transient1 data
abData	UINT8[]		Data of transient frame

Table 29: CCLIES_IF_CMD_SEND_TRANSIENT1_REQ – Send Transient1 request

Packet structure reference

```
typedef struct CCLIES_IF_TRANSIENT1_DATA_Ttag
{
  uint16_t usNetworkID;
  uint16_t usNodeID;
  uint8_t bDataClassification;
  uint16_t usDataSubClassification;
  uint16_t usTransientDataAllSize;
  uint8_t abData[1024];
} CCLIES_IF_TRANSIENT1_DATA_T;

typedef struct CCLIES_IF_SEND_TRANSIENT1_REQ_Ttag
{
  TLR_PACKET_HEADER_T         tHead;
  CCLIES_IF_TRANSIENT1_DATA_T tData;
} CCLIES_IF_SEND_TRANSIENT1_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA745	CCLIES_IF_CMD_SEND_TRANSIENT1_CNF
ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
ulRout	UINT32	X	Routing, do not touch

Table 30: CCLIES_IF_CMD_SEND_TRANSIENT1_CNF – Send Transient1 confirmation

Packet structure reference

```
typedef struct CCLIES_IF_SEND_TRANSIENT1_CNF_Ttag
{
  TLR_PACKET_HEADER_T                    tHead;
} CCLIES_IF_SEND_TRANSIENT1_CNF_T;
```

4.6.2 Transient2

The Transient2 registration allows adding protocols not being addressed by CC-Link IE Field Slave stack.

4.6.2.1 Register for transient2 received indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	2	Packet Data Length in bytes
ulCmd	UINT32	0xA750	CCLIES_IF_CMD_REGISTER_TRANSIENT2_REQ
Data			
bDataClassification	UINT8		
bCommandType	UINT8		

Packet structure reference

```
typedef struct CCLIES_IF_REGISTER_TRANSIENT2_REQ_DATA_Ttag
{
  uint8_t bDataClassification;
  uint8_t bCommandType;
} CCLIES_IF_REGISTER_TRANSIENT2_REQ_DATA_T;

typedef struct CCLIES_IF_REGISTER_TRANSIENT2_REQ_Ttag
{
  TLR_PACKET_HEADER_T                       tHead;
  CCLIES_IF_REGISTER_TRANSIENT2_REQ_DATA_T tData;
} CCLIES_IF_REGISTER_TRANSIENT2_REQ_T;
```

Table 31: CCLIES_IF_CMD_REGISTER_TRANSIENT2_REQ – Register for Transient2 request

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	6	Packet Data Length in bytes
ulCmd	UINT32	0xA751	CCLIES_IF_CMD_REGISTER_TRANSIENT2_CNF
Data			
bDataClassification	UINT8		Same value as in request
bCommandType	UINT8		Same value as in request
ulTransient2Handle	UINT32		

Table 32: CCLIES_IF_CMD_REGISTER_TRANSIENT2_CNF – Register for Transient2 confirmation

Packet structure reference

```
typedef struct CCLIES_IF_REGISTER_TRANSIENT2_CNF_DATA_Ttag
{
  uint8_t bDataClassification;
  uint8_t bCommandType;
  uint32_t ulTransient2Handle;
} CCLIES_IF_REGISTER_TRANSIENT2_CNF_DATA_T;

typedef struct CCLIES_IF_REGISTER_TRANSIENT2_CNF_Ttag
{
  TLR_PACKET_HEADER_T                      tHead;
  CCLIES_IF_REGISTER_TRANSIENT2_CNF_DATA_T tData;
} CCLIES_IF_REGISTER_TRANSIENT2_CNF_T;
```

4.6.2.2 Unregister from Transient2 received indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	4	Packet Data Length in bytes
ulCmd	UINT32	0xA752	CCLIES_IF_CMD_UNREGISTER_TRANSIENT2_REQ
Data			
ulTransient2Handle	UINT32		

Table 33: CCLIES_IF_CMD_UNREGISTER_TRANSIENT2_REQ – Unregister from Transient2 request

Packet structure reference

```
typedef struct CCLIES_IF_UNREGISTER_TRANSIENT2_REQ_DATA_Ttag
{
  uint32_t ulTransient2Handle;
} CCLIES_IF_UNREGISTER_TRANSIENT2_REQ_DATA_T;

typedef struct CCLIES_IF_UNREGISTER_TRANSIENT2_REQ_Ttag
{
  TLR_PACKET_HEADER_T                          tHead;
  CCLIES_IF_UNREGISTER_TRANSIENT2_REQ_DATA_T tData;
} CCLIES_IF_UNREGISTER_TRANSIENT2_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulId	UINT32	0 … 2^{32}-1	Packet Identification as unique number generated by the Source Process of the Packet
ulSta	UINT32		See section Status/Error codes
ulCmd	UINT32	0xA753	CCLIES_IF_CMD_UNREGISTER_TRANSIENT2_CNF – Command

Table 34: CCLIES_IF_CMD_UNREGISTER_TRANSIENT2_CNF – Unregister from Transient2 confirmation

Packet structure reference

```
typedef struct CCLIES_IF_UNREGISTER_TRANSIENT2_CNF_Ttag
{
  TLR_PACKET_HEADER_T                       tHead;
} CCLIES_IF_UNREGISTER_TRANSIENT2_CNF_T;
```

4.6.2.3 Receive Transient2 indication

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	27 + n	Packet Data Length in bytes
ulCmd	UINT32	0xA756	CCLIES_IF_CMD_RECV_TRANSIENT2_IND
Data			
bDataClassification	UINT8		Transient2 data classification
tCclHead.usL	UINT16		Length from bFNO including content in abData
tCclHead.bRsv	UINT8		Set to zero
tCclHead.bTP_SF	UINT8		
tCclHead.bFNO	UINT8		
tCclHead.bDT	UINT8		
tCclHead.bDT	UINT8		
tCclHead.bDA	UINT8		
tCclHead.bSA	UINT8		
tCclHead.bDAT	UINT8		
tCclHead.bSAT	UINT8		
tCclHead.bDMF	UINT8		
tCclHead.bSMF	UINT8		
tCclHead.bDNA	UINT8		
tCclHead.bDS	UINT8		
tCclHead.usDID	UINT16		
tCclHead.bSNA	UINT8		
tCclHead.bSS	UINT8		
tCclHead.usSID	UINT16		
tCclHead.usL1	UINT16		Length from bCT including content in abData
tCclHead.bCT	UINT8		
tCclHead.bRsv2	UINT8		Set to zero
tCclHead.usAPS	UINT16		
abData	UINT8[]		Data of Transient2 frame

Table 35: CCLIES_IF_CMD_RECV_TRANSIENT2_IND – Receive Transient2 indication

Packet structure reference

```
typedef struct CCLIES_IF_RECV_TRANSIENT2_IND_Ttag
{
  TLR_PACKET_HEADER_T         tHead;
  CCLIES_IF_TRANSIENT2_DATA_T tData;
} CCLIES_IF_RECV_TRANSIENT2_IND_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA757	CCLIES_IF_CMD_RECV_TRANSIENT2_RES

Table 36: CCLIES_IF_CMD_RECV_TRANSIENT2_RES – Receive Transient2 response

Packet structure reference

```
typedef struct CCLIES_IF_RECV_TRANSIENT2_RES_Ttag
{
  TLR_PACKET_HEADER_T                       tHead;
} CCLIES_IF_RECV_TRANSIENT2_RES_T;
```

4.6.2.4 Send Transient2 frame

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	27 + n	Packet Data Length in bytes
ulCmd	UINT32	0xA754	CCLIES_IF_CMD_SEND_TRANSIENT2_REQ
Data			
bDataClassification	UINT8		Transient2 data classification
tCclHead.usL	UINT16		Length from bFNO including content in abData
tCclHead.bRsv	UINT8		Set to zero
tCclHead.bTP_SF	UINT8		
tCclHead.bFNO	UINT8		
tCclHead.bDT	UINT8		
tCclHead.bDA	UINT8		
tCclHead.bSA	UINT8		
tCclHead.bDAT	UINT8		
tCclHead.bSAT	UINT8		
tCclHead.bDMF	UINT8		
tCclHead.bSMF	UINT8		
tCclHead.bDNA	UINT8		
tCclHead.bDS	UINT8		
tCclHead.usDID	UINT16		
tCclHead.bSNA	UINT8		
tCclHead.bSS	UINT8		
tCclHead.usSID	UINT16		
tCclHead.usL1	UINT16		Length from bCT including content in abData
tCclHead.bCT	UINT8		
tCclHead.bRsv2	UINT8		Set to zero
tCclHead.usAPS	UINT16		
abData	UINT8[]		Data of Transient2 frame

Table 37: CCLIES_IF_CMD_SEND_TRANSIENT2_REQ – Send Transient2 request

Packet structure reference

```
typedef struct CCLIES_IF_TRANSIENT2_DATA_HEAD_Ttag
{
  uint16_t usL;
  uint8_t bRsv;
  uint8_t bTP_SF;
  uint8_t bFNO;
  uint8_t bDT;
  uint8_t bDA;
  uint8_t bSA;
  uint8_t bDAT;
  uint8_t bSAT;
  uint8_t bDMF;
  uint8_t bSMF;
  uint8_t bDNA;
  uint8_t bDS;
  uint16_t usDID;
  uint8_t bSNA;
  uint8_t bSS;
  uint16_t usSID;
  uint16_t usL1;
  uint8_t bCT;
  uint8_t bRsv2;
  uint16_t usAPS;
} CCLIES_IF_TRANSIENT2_DATA_HEAD_T;

typedef struct CCLIES_IF_TRANSIENT2_DATA_Ttag
{
  uint8_t bDataClassification;
  CCLIES_IF_TRANSIENT2_DATA_HEAD_T tCclHead;
  uint8_t abData[960];
} CCLIES_IF_TRANSIENT2_DATA_T;

typedef struct CCLIES_IF_SEND_TRANSIENT2_REQ_Ttag
{
  TLR_PACKET_HEADER_T         tHead;
  CCLIES_IF_TRANSIENT2_DATA_T tData;
} CCLIES_IF_SEND_TRANSIENT2_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA755	CCLIES_IF_CMD_SEND_TRANSIENT2_CNF

Table 38: CCLIES_IF_CMD_SEND_TRANSIENT2_CNF – Send Transient2 confirmation

Packet structure reference

```
typedef struct CCLIES_IF_SEND_TRANSIENT2_CNF_Ttag
{
  TLR_PACKET_HEADER_T                     tHead;
} CCLIES_IF_SEND_TRANSIENT2_CNF_T;
```

4.6.3 SLMP

The SLMP registration allows adding SLMP requests not being implemented by CC-Link IE Field Slave stack.

The following SLMP request types cannot be registered and are always handled by stack.

Command	Subcommand	Meaning
0x3119	0x0000	SelectInfo request
0x3040	0x0000	Communication test
0x3050	0x0000	Cable test

Table 39: CC-Link IE Field pre-registered SLMP commands/subcommands

4.6.3.1 Register for SLMP received indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	3	Packet Data Length in bytes
ulCmd	UINT32	0xA760	CCLIES_IF_CMD_REGISTER_SLMP_REQ
Data			
usCommand	UINT16		
usSubCommand	UINT16		

Table 40: CCLIES_IF_CMD_REGISTER_SLMP_REQ – Register for SLMP request

Packet structure reference

```
typedef struct CCLIES_IF_REGISTER_SLMP_REQ_DATA_Ttag
{
  uint16_t usCommand;
  uint16_t usSubCommand;
} CCLIES_IF_REGISTER_SLMP_REQ_DATA_T;

typedef struct CCLIES_IF_REGISTER_SLMP_REQ_Ttag
{
  TLR_PACKET_HEADER_T                tHead;
  CCLIES_IF_REGISTER_SLMP_REQ_DATA_T tData;
} CCLIES_IF_REGISTER_SLMP_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	8	Packet Data Length in bytes
ulCmd	UINT32	0xA761	CCLIES_IF_CMD_REGISTER_SLMP_CNF
Data			
usCommand	UINT16		Same value as in request
usSubCommand	UINT16		Same value as in request
ulSlmpHandle	UINT32		

Table 41: CCLIES_IF_CMD_REGISTER_SLMP_CNF – Register for SLMP confirmation

Packet structure reference

```
typedef struct CCLIES_IF_REGISTER_SLMP_CNF_DATA_Ttag
{
  uint16_t usCommand;
  uint16_t usSubCommand;
  uint32_t ulSlmpHandle;
} CCLIES_IF_REGISTER_SLMP_CNF_DATA_T;

typedef struct CCLIES_IF_REGISTER_SLMP_CNF_Ttag
{
  TLR_PACKET_HEADER_T                tHead;
  CCLIES_IF_REGISTER_SLMP_CNF_DATA_T tData;
} CCLIES_IF_REGISTER_SLMP_CNF_T;
```

4.6.3.2 Unregister from SLMP received indications

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	4	Packet Data Length in bytes
ulCmd	UINT32	0xA762	CCLIES_IF_CMD_UNREGISTER_SLMP_REQ
Data			
ulSlmpHandle	UINT32		

Table 42: CCLIES_IF_CMD_UNREGISTER_SLMP_REQ – Unregister from SLMP request

Packet structure reference

```
typedef struct CCLIES_IF_UNREGISTER_SLMP_REQ_DATA_Ttag
{
  uint32_t ulSlmpHandle;
} CCLIES_IF_UNREGISTER_SLMP_REQ_DATA_T;

typedef struct CCLIES_IF_UNREGISTER_SLMP_REQ_Ttag
{
  TLR_PACKET_HEADER_T                  tHead;
  CCLIES_IF_UNREGISTER_SLMP_REQ_DATA_T tData;
} CCLIES_IF_UNREGISTER_SLMP_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA763	CCLIES_IF_CMD_UNREGISTER_SLMP_CNF

Table 43: CCLIES_IF_CMD_UNREGISTER_SLMP_CNF – Unregister from SLMP confirmation

Packet structure reference

```
typedef struct CCLIES_IF_UNREGISTER_SLMP_CNF_Ttag
{
  TLR_PACKET_HEADER_T                    tHead;
} CCLIES_IF_UNREGISTER_SLMP_CNF_T;
```

4.6.3.3 Receive SLMP request indication

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	32 + n	Packet Data Length in bytes
ulCmd	UINT32	0xA766	CCLIES_IF_CMD_RECV_SLMP_IND
Data			
tReq.tCclTransientInfo. bDestAddr	UINT8		
tReq.tCclTransientInfo. bSrcAddr	UINT8		
tReq.tCclTransientInfo. bDestAppType	UINT8		
tReq.tCclTransientInfo. bSrcAppType	UINT8		
tReq.tCclTransientInfo. bDestModuleFlag	UINT8		
tReq.tCclTransientInfo. bSrcModuleFlag	UINT8		
tReq.tCclTransientInfo. usDestID	UINT16		
tReq.tCclTransientInfo. usSourceID	UINT16		
tReq.tCclTransientInfo. bDataNo	UINT8		
tReq.tCclTransientInfo. usAppSeqNo	UINT16		
tReq.tSlmpHead. usFrameType	UINT16	0x0054	SLMP request frame type
tReq.tSlmpHead. usSerialNumber	UINT16		Serial number
tReq.tSlmpHead. usReserved	UINT16	0x0000	
tReq.tSlmpHead. bNetworkNumber	UINT8		
tReq.tSlmpHead. bStationNumber	UINT8		
tReq.tSlmpHead. usModuleIONumber	UINT16		
tReq.tSlmpHead. bRequestingStationNumber	UINT8		

Variable	Type	Value / range	Description
tReq.tSlmpHead.usLength	UINT16	6 + m	Length of data from usTImer including data in abData
usTimer	UINT16		
usCommand	UINT16		SLMP command
usSubCommand	UINT16		SLMP subcommand
abData	UINT8[]		SLMP response data depending on SLMP command and SLMP subcommand. Values according to SLMP specification.

Table 44: `CCLIES_IF_CMD_RECV_SLMP_IND` – Receive SLMP request indication

Packet structure reference

```
typedef struct CCLIES_IF_SLMP_CCL_TRANSIENT_Ttag
{
  uint8_t bDestAddr;
  uint8_t bSrcAddr;
  uint8_t bDestAppType;
  uint8_t bSrcAppType;
  uint8_t bDestModuleFlag;
  uint8_t bSrcModuleFlag;
  uint16_t usDestID;
  uint16_t usSourceID;
  uint8_t bDataNo;
  uint16_t usAppSeqNo;
} CCLIES_IF_SLMP_CCL_TRANSIENT_T;

typedef struct CCLIES_IF_SLMP_HEADER_Ttag
{
  uint16_t usFrameType;
  uint16_t usSerialNumber;
  uint16_t usReserved;
  uint8_t bNetworkNumber;
  uint8_t bStationNumber;
  uint16_t usModuleIONumber;
  uint8_t bRequestingStationNumber;
  uint16_t usLength;
} CCLIES_IF_SLMP_HEADER_T;

typedef struct CCLIES_IF_SLMP_REQUEST_Ttag
{
  CCLIES_IF_SLMP_CCL_TRANSIENT_T tCclTransientInfo;
  CCLIES_IF_SLMP_HEADER_T tSlmpHead;
  uint16_t usTimer;
  uint16_t usCommand;
  uint16_t usSubCommand;
  uint8_t abData[1024];
} CCLIES_IF_SLMP_REQUEST_T;

typedef union CCLIES_IF_SLMP_DATA_Ttag
{
  CCLIES_IF_SLMP_REQUEST_T tReq;
} CCLIES_IF_SLMP_DATA_T;

typedef struct CCLIES_IF_RECV_SLMP_IND_Ttag
{
  TLR_PACKET_HEADER_T   tHead;
  CCLIES_IF_SLMP_DATA_T tData;
} CCLIES_IF_RECV_TRANSIENT1_IND_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA767	CCLIES_IF_CMD_RECV_SLMP_RES

Table 45: CCLIES_IF_CMD_RECV_SLMP_RES – Receive SLMP request response

Packet structure reference

```
typedef struct CCLIES_IF_RECV_SLMP_RES_Ttag
{
  TLR_PACKET_HEADER_T                     tHead;
} CCLIES_IF_RECV_SLMP_RES_T;
```

4.6.3.4 Send SLMP response frame

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32		Packet Data Length in bytes
ulCmd	UINT32	0xA764	CCLIES_IF_CMD_SEND_SLMP_REQ
Data			
tRes.tCclTransientInfo. bDestAddr	UINT8		Value from tReq.tCclTransientInfo.bSrcAddr from SLMP request indication
tRes.tCclTransientInfo. bSrcAddr	UINT8		Value from tReq.tCclTransientInfo.bDestAddr from SLMP request indication
tRes.tCclTransientInfo. bDestAppType	UINT8		Value from tReq.tCclTransientInfo.bSrcAppType from SLMP request indication
tRes.tCclTransientInfo. bSrcAppType	UINT8		Value from tReq.tCclTransientInfo.bDestAppType from SLMP request indication
tRes.tCclTransientInfo. bDestModuleFlag	UINT8		Value from tReq.tCclTransientInfo.bSrcModuleFlag from SLMP request indication
tRes.tCclTransientInfo. bSrcModuleFlag	UINT8		Value from tReq.tCclTransientInfo.bDestModuleFlag from SLMP request indication
tRes.tCclTransientInfo. usDestID	UINT16		Value from tReq.tCclTransientInfo.usSourceID from SLMP request indication
tRes.tCclTransientInfo. usSourceID	UINT16		Value from tReq.tCclTransientInfo.usDestID from SLMP request indication
tRes.tCclTransientInfo. bDataNo	UINT8		Value from tReq.tCclTransientInfo.bDataNo from SLMP request indication
tRes.tCclTransientInfo. usAppSeqNo	UINT16		Value from tReq.tCclTransientInfo.usAppSeqNo from SLMP request indication
tRes.tSlmpHead. usFrameType	UINT16	0x00D4	SLMP response frame type
tRes.tSlmpHead. usSerialNumber	UINT16		Value from tReq.tSlmpHead.usSerialNumber
tRes.tSlmpHead. usReserved	UINT16	0x0000	Value from tReq.tSlmpHead.usReserved
tRes.tSlmpHead. bNetworkNumber	UINT8		Value from tReq.tSlmpHead.bNetworkNumber
tRes.tSlmpHead. bStationNumber	UINT8		Value from tReq.tSlmpHead.bStationNumber
tRes.tSlmpHead. usModuleIONumber	UINT16		Value from tReq.tSlmpHead.usModuleIONumber
tRes.tSlmpHead. bRequestingStationNumber	UINT8		Value from tReq.tSlmpHead.bRequestingStationNumber
tRes.tSlmpHead. usLength	UINT16	2 + m	Length of SLMP response
tRes.usEndCode	UINT16	0	
abData	UINT8[]		SLMP response data

Table 46: CCLIES_IF_CMD_SEND_SLMP_REQ – Send SLMP response request

Packet structure reference

```
typedef struct CCLIES_IF_SLMP_CCL_TRANSIENT_Ttag
{
  uint8_t bDestAddr;
  uint8_t bSrcAddr;
  uint8_t bDestAppType;
  uint8_t bSrcAppType;
  uint8_t bDestModuleFlag;
  uint8_t bSrcModuleFlag;
  uint16_t usDestID;
  uint16_t usSourceID;
  uint8_t bDataNo;
  uint16_t usAppSeqNo;
} CCLIES_IF_SLMP_CCL_TRANSIENT_T;

typedef struct CCLIES_IF_SLMP_HEADER_Ttag
{
  uint16_t usFrameType;
  uint16_t usSerialNumber;
  uint16_t usReserved;
  uint8_t bNetworkNumber;
  uint8_t bStationNumber;
  uint16_t usModuleIONumber;
  uint8_t bRequestingStationNumber;
  uint16_t usLength;
} CCLIES_IF_SLMP_HEADER_T;

Typedef struct CCLIES_IF_SLMP_RESPONSE_Ttag
{
  CCLIES_IF_SLMP_CCL_TRANSIENT_T tCclTransientInfo;
  CCLIES_IF_SLMP_HEADER_T tSlmpHead;
  uint16_t usEndCode;
  uint8_t abData[1024];
} CCLIES_IF_SLMP_RESPONSE_T;

typedef union CCLIES_IF_SLMP_DATA_Ttag
{
  CCLIES_IF_SLMP_RESPONSE_T tRes;
} CCLIES_IF_SLMP_DATA_T;

typedef struct CCLIES_IF_SEND_SLMP_REQ_Ttag
{
  TLR_PACKET_HEADER_T tHead;
  CCLIES_IF_SLMP_DATA_T tData;
} CCLIES_IF_SEND_SLMP_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA745	CCLIES_IF_CMD_SEND_SLMP_CNF

Table 47: CCLIES_IF_CMD_SEND_SLMP_CNF – Send SLMP response confirmation

Packet structure reference

```
typedef struct CCLIES_IF_SEND_SLMP_CNF_Ttag
{
  TLR_PACKET_HEADER_T                    tHead;
} CCLIES_IF_SEND_SLMP_CNF_T;
```

4.6.3.5 Send SLMP error response frame

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	37	Packet Data Length in bytes
ulCmd	UINT32	0xA764	CCLIES_IF_CMD_SEND_SLMP_REQ
tData			
tErrorRes. tCclTransientInfo. bDestAddr	UINT8		Value from tReq.tCclTransientInfo.bSrcAddr from SLMP request indication
tErrorRes. tCclTransientInfo. bSrcAddr	UINT8		Value from tReq.tCclTransientInfo.bDestAddr from SLMP request indication
tErrorRes. tCclTransientInfo. bDestAppType	UINT8		Value from tReq.tCclTransientInfo.bSrcAppType from SLMP request indication
tErrorRes. tCclTransientInfo. bSrcAppType	UINT8		Value from tReq.tCclTransientInfo.bDestAppType from SLMP request indication
tErrorRes. tCclTransientInfo. bDestModuleFlag	UINT8		Value from tReq.tCclTransientInfo.bSrcModuleFlag from SLMP request indication
tErrorRes. tCclTransientInfo. bSrcModuleFlag	UINT8		Value from tReq.tCclTransientInfo.bDestModuleFlag from SLMP request indication
tErrorRes. tCclTransientInfo. usDestID	UINT16		Value from tReq.tCclTransientInfo.usSourceID from SLMP request indication
tErrorRes. tCclTransientInfo. usSourceID	UINT16		Value from tReq.tCclTransientInfo.usDestID from SLMP request indication
tErrorRes. tCclTransientInfo. bDataNo	UINT8		Value from tReq.tCclTransientInfo.bDataNo from SLMP request indication
tErrorRes. tCclTransientInfo. usAppSeqNo	UINT16		Value from tReq.tCclTransientInfo.usAppSeqNo from SLMP request indication
tErrorRes. tSlmpHead. usFrameType	UINT16	0x00D4	SLMP response frame type
tErrorRes.tSlmpHead. usSerialNumber	UINT16		Value from tReq.tSlmpHead.usSerialNumber
tErrorRes.tSlmpHead. usReserved	UINT16		Value from tReq.tSlmpHead.usReserved
tErrorRes.tSlmpHead. bNetworkNumber	UINT8		Value from tReq.tSlmpHead.bNetworkNumber
tErrorRes.tSlmpHead. bStationNumber	UINT8		Value from tReq.tSlmpHead.bStationNumber
tErrorRes.tSlmpHead. usModuleIONumber	UINT16		Value from tReq.tSlmpHead.usModuleIONumber
tErrorRes.tSlmpHead. bRequestingStationNumber	UINT8		Value from tReq.tSlmpHead.bRequestingStationNumber
tErrorRes.tSlmpHead. usLength	UINT16	11	

Variable	Type	Value / range	Description
tErrorRes.usEndCode	UINT16	1 … 0xFFFF	
tErrorRes.tError. bNetworkNo	UINT8		Automatically filled in by stack
tErrorRes.tError. bStationNo	UINT8		Automatically filled in by stack
tErrorRes.tError. usRespUtIONo	UINT16	0x3FF	
tErrorRes.tError. bRequStNo	UINT8	0	
tErrorRes.tError. usCmd	UINT16		Command value from request indication
tErrorRes.tError. usSubCmd	UINT16		Subcommand value from request indication

Table 48: `CCLIES_IF_CMD_SEND_SLMP_REQ` – Send SLMP response request (error response)

Packet structure reference

```
typedef struct CCLIES_IF_SLMP_CCL_TRANSIENT_Ttag
{
  uint8_t bDestAddr;
  uint8_t bSrcAddr;
  uint8_t bDestAppType;
  uint8_t bSrcAppType;
  uint8_t bDestModuleFlag;
  uint8_t bSrcModuleFlag;
  uint16_t usDestID;
  uint16_t usSourceID;
  uint8_t bDataNo;
  uint16_t usAppSeqNo;
} CCLIES_IF_SLMP_CCL_TRANSIENT_T;

typedef struct CCLIES_IF_SLMP_HEADER_Ttag
{
  uint16_t usFrameType;
  uint16_t usSerialNumber;
  uint16_t usReserved;
  uint8_t bNetworkNumber;
  uint8_t bStationNumber;
  uint16_t usModuleIONumber;
  uint8_t bRequestingStationNumber;
  uint16_t usLength;
} CCLIES_IF_SLMP_HEADER_T;

Typedef struct CCLIES_IF_SLMP_RESPONSE_Ttag
{
  CCLIES_IF_SLMP_CCL_TRANSIENT_T tCclTransientInfo;
  CCLIES_IF_SLMP_HEADER_T tSlmpHead;
  uint16_t usEndCode;
  struct
  {
    uint8_t bNetworkNo;
    uint8_t bStationNo;
    uint16_t usRespUtIONo;
    uint8_t bRequStNo;
    uint16_t usCmd;
    uint16_t usSubCmd;
  } tError;
} CCLIES_IF_SLMP_ERROR_RESPONSE_T;

typedef union CCLIES_IF_SLMP_DATA_Ttag
{
  CCLIES_IF_SLMP_ERROR_RESPONSE_T tErrorRes;
} CCLIES_IF_SLMP_DATA_T;
```

```
typedef struct CCLIES_IF_SEND_SLMP_REQ_Ttag
{
  TLR_PACKET_HEADER_T tHead;
  CCLIES_IF_SLMP_DATA_T tData;
} CCLIES_IF_SEND_SLMP_REQ_T;
```

Packet description

Variable	Type	Value / range	Description
ulDest	UINT32		Destination queue-handle
ulLen	UINT32	0	Packet Data Length in bytes
ulCmd	UINT32	0xA745	CCLIES_IF_CMD_SEND_SLMP_CNF

Table 49: CCLIES_IF_CMD_SEND_SLMP_CNF – Send SLMP response confirmation

Packet structure reference

```
typedef struct CCLIES_IF_SEND_SLMP_CNF_Ttag
{
  TLR_PACKET_HEADER_T                    tHead;
} CCLIES_IF_SEND_SLMP_CNF_T;
```

5 Status/Error codes

5.1 Error codes: CC-Link IE Field Slave

Hexadecimal value	Definition and description
0xC0F30001	ERR_CCLIES_REQUEST_DESTINATION_PROBLEM Request destination problem.
0xC0F30002	ERR_CCLIES_TRANSIENT1_ALREADY_REGISTERED Transient1 already registered.
0xC0F30003	ERR_CCLIES_TRANSIENT1_NOT_REGISTERED Transient1 not registered.
0xC0F30004	ERR_CCLIES_TRANSIENT2_ALREADY_REGISTERED Transient2 already registered.
0xC0F30005	ERR_CCLIES_TRANSIENT2_NOT_REGISTERED Transient2 not registered.
0xC0F30006	ERR_CCLIES_SLMP_ALREADY_REGISTERED SLMP already registered.
0xC0F30007	ERR_CCLIES_SLMP_NOT_REGISTERED SLMP not registered.
0xC0F30008	ERR_CCLIES_SLMP_INVALID_TYPE_PARAMETER Invalid SLMP Type parameter.
0xC0F30009	ERR_CCLIES_CONFIGURATION_BUSY Configuration busy.
0xC0F3000A	ERR_CCLIES_INVALID_DEVICE_TYPE Invalid device type.
0xC0F3000B	ERR_CCLIES_INVALID_MAX_RX_BYTES Invalid maximum RX bytes.
0xC0F3000C	ERR_CCLIES_INVALID_MAX_RY_BYTES Invalid maximum RY bytes.
0xC0F3000D	ERR_CCLIES_INVALID_MAX_RWR_POINTS Invalid maximum RWr points.
0xC0F3000E	ERR_CCLIES_INVALID_MAX_RWW_POINTS Invalid maximum RWw points.
0xC0F3000F	ERR_CCLIES_INVALID_MIN_RX_BYTES Invalid minimum RX bytes.
0xC0F30010	ERR_CCLIES_INVALID_MIN_RY_BYTES Invalid minimum RY bytes.
0xC0F30011	ERR_CCLIES_INVALID_MIN_RWR_POINTS Invalid minimum RWr points.
0xC0F30012	ERR_CCLIES_INVALID_MIN_RWW_POINTS Invalid minimum RWw points.
0xC0F30013	ERR_CCLIES_INVALID_IOTYPE Invalid IO-Type.
0xC0F30014	ERR_CCLIES_INVALID_NET_NUMBER Invalid network number.
0xC0F30015	ERR_CCLIES_CYCLIC_STOPPED Cyclic stopped.

Hexadecimal value	Definition and description
0xC0F30016	ERR_CCLIES_ERROR_BLINK Error Blink.
0xC0F30017	ERR_CCLIES_INVALID_STATION_NUMBER Invalid station number.
0xC0F3F000	ERR_CCLIES_LLD_INVALID_FIRMWARE LLD: Invalid firmware.
0xC0F3F001	ERR_CCLIES_LLD_NOT_READY_COOKIE LLD: Not ready cookie.
0xC0F3F002	ERR_CCLIES_LLD_NOT_READY_STATUS LLD: Not ready status.
0xC0F3F003	ERR_CCLIES_LLD_INVALID_PARAMETER LLD: Invalid parameter.
0xC0F3F004	ERR_CCLIES_LLD_UNKNOWN_ERROR LLD: Unknown error.
0xC0F3F005	ERR_CCLIES_LLD_OUT_OF_MEMORY LLD: Out of memory.
0xC0F3F006	ERR_CCLIES_LLD_IRQ_INIT_ERROR LLD: IRQ init error.
0xC0F3F007	ERR_CCLIES_LLD_BUSY LLD: Busy.

Table 50: Status/error codes: CC-Link IE Field Slave

5.2 Status/error codes: CC-Link IE Field SLMP Endcodes

Hexadecimal value	Definition and description
0xC0F6C059	ERR_SLMP_ENDCODE_CMD_SUBCMD_ERROR Command/Subcommand error.
0xC0F6C05C	ERR_SLMP_ENDCODE_ERROR_IN_REQ_MESSAGE Error in request message.
0xC0F6C061	ERR_SLMP_ENDCODE_REQ_LENGTH_DOES_NOT_MATCH Request length does not match.
0xC0F6CEE0	ERR_SLMP_ENDCODE_BUSY Busy.
0xC0F6CEE1	ERR_SLMP_ENDCODE_REQ_SIZE_EXCEEDED_EFFECT_PROCESS_RANGE Request size exceeded effective processing range.
0xC0F6CEE2	ERR_SLMP_ENDCODE_RES_SIZE_EXCEEDED_EFFECT_PROCESS_RANGE Response size exceeded effective processing range.
0xC0F6CF10	ERR_SLMP_ENDCODE_SPECIFIED_SERVER_INFO_NO_NOT_EXIST Specified server info number does not exist.
0xC0F6CF20	ERR_SLMP_ENDCODE_CONTAINED_ITEMS_CANNOT_BE_SET Contained items cannot be set.
0xC0F6CF30	ERR_SLMP_ENDCODE_PARAM_ID_DOES_NOT_EXIST Parameter id does not exist.
0xC0F6CF31	ERR_SLMP_ENDCODE_WRITE_EXCLUSIVE_START_NOT_PERFORMED Write exclusive start not performed.
0xC0F6CF70	ERR_SLMP_ENDCODE_RELAY_PATH_DEST_COMM_ERROR Relay path destination communication error.
0xC0F6CF71	ERR_SLMP_ENDCODE_TIMEOUT_OCCURED Timeout occured.

Table 51: Status/error codes: CC-Link IE Field SLMP Endcodes

6 Appendix

6.1 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.

6.2 List of tables

Table 1: List of Revisions 3
Table 2: Terms, Abbreviations and Definitions 5
Table 3: References to documents 5
Table 4: LED states for the CC-Link IE Field Slave protocol 8
Table 5: LED Names CC-Link IE Field Slave protocol 8
Table 6: CCLIES_IF_CMD_SET_CONFIG_REQ - Set Configuration request 10
Table 7: CCLIES_IF_CMD_SET_CONFIG_CNF - Set Configuration confirmation 11
Table 8: RCX_REGISTER_APP_REQ – Register for status indications request 13
Table 9: RCX_REGISTER_APP_CNF - Register for status indications confirmation 13
Table 10: RCX_UNREGISTER_APP_REQ – Unregister from status indications request 14
Table 11: RCX_UNREGISTER_APP_CNF - Unregister from status indications confirmation 14
Table 12: CCLIES_IF_CMD_STATUS_IND – Status indication 15
Table 13: Detailed Application Operation Status 16
Table 14: Error Detection Status 17
Table 15: CCLIES_IF_CMD_STATUS_RES – Status response 17
Table 16: CCLIES_IF_CMD_GET_STATUS_REQ – Get status request 18
Table 17: CCLIES_IF_CMD_GET_STATUS_CNF – Get status confirmation 19
Table 18: Detailed Application Operation Status 20
Table 19: Error Detection Status 21
Table 20: CCLIES_IF_CMD_SET_DET_NODE_STATUS_REQ – Set detailed node status request 22
Table 21: CCLIES_IF_CMD_SET_DET_NODE_STATUS_CNF – Set detected node status confirmation 23
Table 22: CC-Link IE Field pre-registered Transient1 protocols 24
Table 23: CCLIES_IF_CMD_REGISTER_TRANSIENT1_REQ - Register for Transient1 request 24
Table 24: CCLIES_IF_CMD_REGISTER_TRANSIENT1_CNF – Register for Transient1 confirmation 25
Table 25: CCLIES_IF_CMD_UNREGISTER_TRANSIENT1_REQ – Unregister from Transient1 request 26
Table 26: CCLIES_IF_CMD_UNREGISTER_TRANSIENT1_CNF – Unregister from Transient1 confirmation 26
Table 27: CCLIES_IF_CMD_RECV_TRANSIENT1_IND – Receive Transient1 indication 27
Table 28: CCLIES_IF_CMD_RECV_TRANSIENT1_RES – Receive Transient1 response 27
Table 29: CCLIES_IF_CMD_SEND_TRANSIENT1_REQ – Send Transient1 request 28
Table 30: CCLIES_IF_CMD_SEND_TRANSIENT1_CNF – Send Transient1 confirmation 29
Table 31: CCLIES_IF_CMD_REGISTER_TRANSIENT2_REQ – Register for Transient2 request 30
Table 32: CCLIES_IF_CMD_REGISTER_TRANSIENT2_CNF – Register for Transient2 confirmation 31
Table 33: CCLIES_IF_CMD_UNREGISTER_TRANSIENT2_REQ – Unregister from Transient2 request 32
Table 34: CCLIES_IF_CMD_UNREGISTER_TRANSIENT2_CNF – Unregister from Transient2 confirmation 32
Table 35: CCLIES_IF_CMD_RECV_TRANSIENT2_IND – Receive Transient2 indication 33
Table 36: CCLIES_IF_CMD_RECV_TRANSIENT2_RES – Receive Transient2 response 34
Table 37: CCLIES_IF_CMD_SEND_TRANSIENT2_REQ – Send Transient2 request 35
Table 38: CCLIES_IF_CMD_SEND_TRANSIENT2_CNF – Send Transient2 confirmation 36
Table 39: CC-Link IE Field pre-registered SLMP commands/subcommands 37
Table 40: CCLIES_IF_CMD_REGISTER_SLMP_REQ – Register for SLMP request 37
Table 41: CCLIES_IF_CMD_REGISTER_SLMP_CNF – Register for SLMP confirmation 38
Table 42: CCLIES_IF_CMD_UNREGISTER_SLMP_REQ – Unregister from SLMP request 39
Table 43: CCLIES_IF_CMD_UNREGISTER_SLMP_CNF – Unregister from SLMP confirmation 39
Table 44: CCLIES_IF_CMD_RECV_SLMP_IND – Receive SLMP request indication 41
Table 45: CCLIES_IF_CMD_RECV_SLMP_RES – Receive SLMP request response 42
Table 46: CCLIES_IF_CMD_SEND_SLMP_REQ – Send SLMP response request 43
Table 47: CCLIES_IF_CMD_SEND_SLMP_CNF – Send SLMP response confirmation 44
Table 48: CCLIES_IF_CMD_SEND_SLMP_REQ – Send SLMP response request (error response) 46
Table 49: CCLIES_IF_CMD_SEND_SLMP_CNF – Send SLMP response confirmation 47
Table 50: Status/error codes: CC-Link IE Field Slave 49
Table 51: Status/error codes: CC-Link IE Field SLMP Endcodes 50

6.3 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