GOAL V2.21.1

CC-Link IE Field Basic
User Manual
port GmbH, 2020

port GmbH
Regensburger Str. 7
D-06132 Halle/Saale

Disclaimer

This manual represents the current state of the product. Please check with port.de for the latest version as the document may have a newer version since errors may be corrected or changes for a newer version of the product may be incorporated. Port.de assumes no responsibility for errors in this document. Qualified feedback is appreciated at service@port.de.

This document is the Intellectual Property of port.de and is intended to be used with the described product only. It may be forwarded and/or copied in the original and unmodified format. All rights reserved.

The product enables to use technologies such as PROFINET, EtherNet/IP and/or EtherCAT and others. These technologies are promoted by trade organizations, such as PNO (profibus.org), ODVA (odva.org) or ETG (ethercat.org). These trade organizations as well maintain the specification and care about legal issues. We strongly recommend to become a member of these organisations. Most technologies are making use of patented or otherwise copyrighted technologies, approaches or other intellectual property. The membership usually automatically entitles the member for use of most of the technology-inherent copyrighted or otherwise protected Intellectual Property of the corresponding trade organization and most 3rd parties. Otherwise the user will need to obtain licenses for many patented technologies separately.

Further we suggest to you to subscribe to the corresponding Conformance Test Tool of these trade organizations. For instance the ODVA only accepts conformance test applications from companies who have a valid membership and have a valid subscription to the recent Conformance Test Tool. We as port are members in all corresponding organizations and are holding a subscription to these tools - however you as a customer need to have an own membership and an own subscription to the tool.

All rights reserved

The programs, boards and documentations supplied by port GmbH are created with due diligence, checked carefully and tested on several applications. Nevertheless, port GmbH cannot guarantee and nor assume liability that the program, the hardware board or the documentation are error-free or appropriate to serve a specific customer purpose. In particular performance characteristics and technical data given in this document may not be interpreted to be guaranteed product features in any legal sense.

For consequential damages, every legal responsibility or liability is excluded. port has the right to modify the products described or their documentation at any time without prior warning, as long as these changes are made for reasons of reliability or technical improvement. All rights of this documentation are with port. Unless expressively granted - the transfer of rights to third parties or duplication of this document in any form, whole or in part, is subject to written approval by port. Copies of this document may however be made exclusively for the use of the user and his engineers. The user is thereby responsible that third parties do not obtain access to these copies. The soft- and hardware designations used are mostly registered and are subject to copyright.

Regensburger Straße 7
D-06132 Halle
Tel. +49 345 - 777 55 0
Fax. +49 345 - 777 55 20
E-Mail service@port.de
www.port.de
www.port-automation.com

Contents

List of Tables

1 General Information

1.1 Changelog

Table 1: Changelog

Date	Revision	Authors	Comment
2020-09-17	1.0	mhr	Initial revision.

1.2 Authors

The following authors have been working on this document.

Table 2: Authors of the GOAL Time Synchronization User Manual

Sign	Name	Mail
mhr	Martin Herberg	mhr@port.de

2 Introduction to CC-Link IE Field Basic

This document describes the usage of a CC-Link IE Field Basic Slave Protocol Stack for GOAL. CC-Link IE Field Basic is abbreviated to CCLIEFB throughout this document.

2.1 Slave Stations

Using the GOAL CCLIEFB slave stack it is possible to setup a slave occupying 1 to 16 stations. Each station can be interpreted as a physical or virtual participant of the CCLIEFB network. To each of the stations there is a fixed amount of I/O data with fixed size, called Link Devices. I.e. to exchange more I/O data the slave may occupy more stations.

Table 3: Link Devices per station.

Link Device	Type	Size	Master access	Slave access
RY	Bit Output	64 bits (8 bytes)	write	read
RX	Bit Input	64 bits (8 bytes)	read	write
RWw	Word Output	32 words (64 bytes)	write	read
RWr	Word Input	32 words (64 bytes)	read	write

This leads to 72 bytes process data for each direction per slave, leading up to a maximum of $16 \cdot 72 = 1152$ bytes per direction for a GOAl CCLIEFB slave.

For a slave containing more than one station the link devices sizes are multiples of the given sizes. E.g. a slave occupying 2 stations has 16 bytes RY data.

The meaning of each bit within the Link Devices is application specific, i.e. the protocol specifies the size of each Link device but the application decides which data points are actually used.

2.2 CSP Plus File

The CSP Plus file is the device description needed by a PLC to identify and to communication with a device. A sample is given *appl/goal_ccliefb* . The device is identified completely by the fields “VendorCode”, “ProductID” and “Version”. The productID is also known as Model Code. These values must be the same as in the configuration of the GOAL CCSLIEFB slave stack. Attention: The product ID in the CSP Plus file is a string, and must correspond to the configured product ID as a hexadecimal value.

3 Application Programming Interface

This chapter lists the API functions that are provided by GOAL CCLIEFB. See the applications provides in *appl/-goal_ccliefb* for a demonstration. For further details please refer to the Reference Manual.

The function goal_ccIIeFbNew creates a handle that represents the stack instance. All functions with the exception of goal_ccIIeFbInit and the configuration functions require this handle as a parameter.

In order to use these functions the header *goal_ccl_ie_fb.h* must be included.

3.1 goal_ccIIeFbInit

Register the GOAL CC-Link IE Field Basic stack in GOAL. Returns a GOAL_STATUS_T value as result. This function must be called within the function appl_init().

No parameters required.

```
1 res = goal_ccIIeFbInit();
```

3.2 goal_ccIIeFbNew

Create a new GOAL CC-Link IE Field Basic stack slave instance with the given *id* and register an application callback handler. Returns a GOAL_STATUS_T value as result. This function must be called within the function appl_setup().

Table 4: goal_ccIIeFbNew parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T **ppCfb	GOAL CCLIEFB instance reference
uint32_t id	instance id
GOAL_CCLIEFB_FUNC_CB_T pFunc	GOAL CCLIEFB callback function

```
1 res = goal_ccIIeFbNew(&pCfbHdl, GOAL_CCLIEFB_INSTANCE_DEFAULT, appl_cclfbCb);
```

3.3 goal_ccIIeFbAppStopSet

This function is used to indicate the stop status to the Master station. If the application is stopped process data is neither sent nor received until the application is started again. Here a GOAL_BOOL_T flag is passed, GOAL_TRUE means the application is stopping, GOAL_FALSE means, that the application is starting (again). This flag is sent in the cyclicData response messages of the slave. Returns a GOAL_STATUS_T value as result.

Table 5: goal_ccIIeFbAppStopSet parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle
GOAL_BOOL_T stop	application stop flag

```
1 res = goal_ccIIeFbAppStopSet(&pCfbHdl, BOOL_TRUE);
```

3.4 goal_ccIIeFbAppErrorCodeSet

This function is used to indicate the stop status to the Master station. This can be used to indicate the reason of an stopped application. These error bytes are sent in the cyclicData response messages of the slave. Returns a GOAL_STATUS_T value as result.

Table 6: goal_ccIIeFbAppErrorCodeSet parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle
uint16_t errCode	application error code
uint32_t errDetails	additional information

```
1 res = goal_ccIIeFbAppErrorCodeSet(&pCfbHdl, 1, 17);
```

3.5 goal_ccIIeFbIoAccessStart

This function is used to get access to the I/O data. This function must be used before any goal_ccIIeFbOutputGet and goal_ccIIeFbInputSet call corresponding to one process data image to lock internal resources to ensure consistency of I/O data across multiple read and write operations. Returns a GOAL_STATUS_T value as result.

Table 7: goal_ccIIeFbIoAccessStart parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle

```
1 res = goal_ccIIeFbIoAccessStart(pCfbHdl);
```

3.6 goal_ccIIeFbIoAccessEnd

This function is used to return access to the I/O data. This function must be used after all goal_ccIIeFbOutputGet and goal_ccIIeFbInputSet calls corresponding to one process data image to unlock internal resources to ensure the correct update of I/O data. Returns a GOAL_STATUS_T value as result.

Table 8: goal_ccIIeFbIoAccessEnd parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle

```
1 res = goal_ccIIeFbIoAccessEnd(pCfbHdl);
```

3.7 goal_ccIIeFbOutputGet

This function is used to get the output data from the master belonging to one of the link devices RY or RWw. It is possible to read more than one station at once. Further it is possible to get a part of the cyclic data only by addressing

using an offset. Between two different read commands it may be possible, that the stack updated the cyclic data. Returns a GOAL_STATUS_T value as result.

Table 9: goal_ccIIeFbOutputGet parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle
GOAL_CCLIEFB_LINK_DEV_ID_T devId	link device Id
uint8_t *pBuf	write buffer
uint16_t bufLen	write buffer length
uint16_t offset	offset within link device

```
uint8_t ry0;                                    /* first byte from RY Link device */
res = goal_ccIIeFbOutputGet(pCfbHdl,
                            GOAL_CCLIEFB_LINK_DEV_RY,
                            &ry0,
                            sizeof(uint8_t),
                            0);
```

3.8 goal_ccIIeFbInputSet

This function is used to get the output data from the master belonging to one of the link devices RX or RWr. It is possible to write more than one station at once. Further it is possible to write a part of the cyclic data only by addressing using an offset. Between two different write commands it may be possible, that the stack sent the new data already. Returns a GOAL_STATUS_T value as result.

Table 10: goal_ccIIeFbInputSet parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle
GOAL_CCLIEFB_LINK_DEV_ID_T devId	link device Id
uint8_t *pBuf	new data
uint16_t bufLen	data length
uint16_t offset	offset within link device

```
uint8_t rx0 = 0;                        /* first byte from RX Link device */
res = goal_ccIIeFbInputSet(pCfbHdl,
                           GOAL_CCLIEFB_LINK_DEV_RX,
                           &rx0,
                           sizeof(uint8_t),
                           0);
```

3.9 goal_ccIIeFbVersionGet

Get the version of the GOAL CCLIEFB stack. Returns a GOAL_STATUS_T value as result.

Table 11: goal_ccIeFbVersionGet parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle
const char *pVersion	version string

```
const char *pVersion;                          /* GOAL CCLIEFB version string */
res = goal_ccIeFbVersionGet(pCfbHdl, &pVersion);
if (GOAL_RES_OK(res)) {
    goal_logInfo("started CC-Link IE Field Basic stack, version %s", pVersion);
}
```

4 Stack Configuration

This chapter list all functions that can be used to configure the GOAL CCLIEFB stack. These functions must be called within appl_setup() before goal_ccIIeFbNew() is called. Te configuration function alter global configuration variables which are copied during goal_ccIIeFbNew to the stack instance data. The configuration influences the behavior of the GOAL CCLIEFB stack and the resource allocation during initialization.

4.1 goal_ccIIeFbCfgDeviceVersionSet

This function the device version of the station. The device version must correspond to the device version in the CSP plus file needed by a PLC to identify the device.

Default Value: 0x0101

Returns a GOAL_STATUS_T value as result.

Table 12: goal_ccIIeFbCfgDeviceVersionSet parameters

Parameter	Description
uint16_t version	new device version

```
1 res = goal_ccIIeFbCfgDeviceVersionSet(0x0001);
```

4.2 goal_ccIIeFbCfgDeviceVendorCodeSet

This function the vendor code of the station. The vendor code must correspond to the vendor ID in the CSP plus file needed by a PLC to identify the device.

Default Value: 0x0E13

Returns a GOAL_STATUS_T value as result.

Table 13: goal_ccIIeFbCfgDeviceVendorCodeSet parameters

Parameter	Description
uint16_t vendorCode	new vendor code

```
1 res = goal_ccIIeFbCfgDeviceVendorCodeSet(0x0E13);
```

4.3 goal_ccIIeFbCfgDeviceModelCodeSet

This function sets the model code of the station. The model code must correspond to the model code in the CSP plus file needed by a PLC to identify the device.

Default Value: 0x00000001

Returns a GOAL_STATUS_T value as result.

Table 14: goal_ccIIeFbCfgDeviceModelCodeSet parameters

Parameter	Description
uint32_t productId	new product Id

```
res = goal_ccIIeFbCfgDeviceModelCodeSet(0x00000001);
```

4.4 goal_ccIIeFbCfgNumStationsSet

This function sets number of occupied Stations of the station. This number of station is the maximum number of stations which may requested by the PLC. This number also determines the total sizes of all four link devices and therefore the size of the process data.

Default Value: 4

Returns a GOAL_STATUS_T value as result.

Table 15: goal_ccIIeFbCfgNumStationsSet parameters

Parameter	Description
uint8_t numStations	number of occupied stations

```
res = goal_ccIIeFbCfgNumStationsSet(2);
```

5 Application callback.

The application should register a callback handler with the function goal_ccIIeFbNew(). This callback will be called by the stack to inform the application about an event. The callback has three parameters:

Table 16: Application callback handler parameters

Parameter	Description
GOAL_CCLIEFB_HANDLE_T *pCfb	GOAL CCLIEFB handle
GOAL_CCLIEFB_CB_ID_T cbId	callback ID
GOAL_CCLIEFB_CB_DATA_T *pCbData	callback data

The callback ID is used to indicate the event. Depending on the event *pCbData* must be casted to the appropriate data type. The callback must return a GOAL_STATUS_T value. This return value is used for certain events as a decision indicator. If no callback handler is registered, a default callback handler is used. It accepts all events.

```
static GOAL_STATUS_T appl_cclfbCb(
    GOAL_CCLIEFB_HANDLE_T *pCfb,                    /**< GOAL CCLIEFB handle */
    GOAL_CCLIEFB_CB_ID_T cbId,                      /**< callback ID */
    GOAL_CCLIEFB_CB_DATA_T *pCbData                 /**< callback data */
)
{
    GOAL_STATUS_T res = GOAL_OK;                    /* result */
    UNUSEDARG(pCfb);

    switch (cbId) {
        /* ... */
    }

    return res;
```

5.1 GOAL_CCLIEFB_CB_ID_TIMEOUT

The connection to Master Station timed out. This callback is called if to many cyclicData request messages of the master are not received if the communication is running.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_TIMEOUT:
    goal_logInfo("connection to Master Station timed out");
    appRun = GOAL_FALSE;
    break;

/* ... */
```

5.2 GOAL_CCLIEFB_CB_ID_CYC_STOP

Received cyclic stop instruction from Master Station. This callback is called if the master stops the cyclic communication.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_CYC_STOP:
    goal_logInfo("cyclic stop instruction from Master Station");
    appRun = GOAL_FALSE;
    break;

/* ... */
```

5.3 GOAL_CCLIEFB_CB_ID_NEW_MASTER

Found and connected to new master station. This callback if the GOAL CCLIEFB slave is not connected to any master and a new master starts sending cyclic request messages.

callback data type: ID of new master station

callback data: uint32_t *

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_NEW_MASTER:
    goal_logInfo("connected to a new Master Station with ID: 0x%08"FMT_x32, *
    pCbData->pNewMasterId);
    appRun = GOAL_TRUE;
    break;

/* ... */
```

5.4 GOAL_CCLIEFB_CB_ID_MASTER_STOP_ERR

Master Station application stopped because of error. This callback is called if the master application stops because of an error (master protocol version 2).

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_MASTER_STOP_USER:
    goal_logInfo("Master Station application stopped because of error");
    appRun = GOAL_FALSE;
    break;

/* ... */
```

5.5 GOAL_CCLIEFB_CB_ID_MASTER_STOP_USER

Master Station application stopped because of error. This callback is called if the master application stops because of the user (master protocol version 2).

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_MASTER_STOP_USER:
    goal_logInfo("Master Station application stopped by user");
    appRun = GOAL_FALSE;
    break;

/* ... */
```

5.6 GOAL_CCLIEFB_CB_ID_MASTER_STOP_UNKNOWN

Master Station application stopped because of error. This callback is called if the master application stops (master protocol version 1).

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_MASTER_STOP_UNKNOWN:
    goal_logInfo("Master Station application stopped");
    appRun = GOAL_FALSE;
    break;

/* ... */
```

5.7 GOAL_CCLIEFB_CB_ID_MASTER_RUN

Master Station application running. This callback is called if a stopped master application is running again.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_MASTER_RUN:
    goal_logInfo("Master Station application running");
    appRun = GOAL_TRUE;
    break;

/* ... */
```

5.8 GOAL_CCLIEFB_CB_ID_CYC_DATA_VALID

The cyclic data from Master Station is valid. Detected that the master changed cyclic transmission state to 1 indicating that cyclic data started to exchange.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_CYC_DATA_VALID:
    goal_logInfo("cyclic data from Master Station is valid");
    break;

/* ... */
```

5.9 GOAL_CCLIEFB_CB_ID_CYC_DATA_INVALID

The cyclic data from Master Station is invalid. Detected that the master changed cyclic transmission state to 0 indicating that cyclic data stopped to exchange.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_CYC_DATA_INVALID:
    goal_logInfo("cyclic data from Master Station is invalid");
    break;

/* ... */
```

5.10 GOAL_CCLIEFB_CB_ID_DUPL_MASTER

The station detected master station duplication. Another master tried to connected during a running connection with the current master.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_DUPL_MASTER:
    goal_logInfo("detected master station duplication");
    break;

/* ... */
```

5.11 GOAL_CCLIEFB_CB_ID_DUPL_SLAVE

The station detected master station duplication. Received a frame with active cyclic connection and same station ID of the device, i.e. the master is exchanging data with another device with the same station ID. Therefore the device knows that its station ID s a duplicate.

callback data type: none

callback data: none

return value: ignored

```
/* ... */

case GOAL_CCLIEFB_CB_ID_DUPL_SLAVE:
    goal_logInfo("detected slave duplication station ID error");
    break;

/* ... */
```