



User Manual
CC-Link IE TSN

***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port.de for the latest version as the document may have a newer version since errors may be corrected or changes for a newer version of the product may be incorporated.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 expressly 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.

Copyright

© 2020 port GmbH

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

1	Master Station SDK	7
1.1	Writing an Application for the Master SDK	8
1.1.1	Overview	8
1.1.2	Configuration of the CC-Link IE TSN protocol stack	8
1.1.3	Creating a new instance of the CC-Link IE TSN protocol stack.....	9
1.1.4	Runtime behavior of the CC-Link IE TSN Master stack	10
1.1.5	Remote Station Configuration.....	11
1.1.6	Starting the CC-Link IE TSN Master Stack.....	13
1.1.7	Functions used during Run phase	13
1.1.8	Application Callback	14
1.1.9	Configuration of CANopen Remote Stations.....	17
1.1.10	CANopen Data Callback.....	18
1.2	Supported Platforms	20
1.2.1	NXP LS1028A	20
1.3	Conformance Test	24
2	Remote Station SDK	25
2.1	Writing an Application for the Remote Station SDK	26
2.1.1	Overview	26
2.1.2	Configuration of the CC-Link IE TSN protocol stack	26
2.1.3	Creating a new instance of the CC-Link IE TSN protocol stack.....	27
2.1.4	Runtime behavior of the CC-Link IE TSN Remote Station stack.....	27
2.1.5	Starting the CC-Link IE TSN Remote Station Stack	28
2.1.6	Functions used during Run phase	28
2.1.7	Application Callback	29
2.1.8	Data Access with CANopen	30
2.2	Supported Platforms	32
2.3	Conformance Test	33
3	ICC support.....	34

List of Figures

Figure 1.1: Creating a new instance of the CC-Link IE TSN master stack	9
Figure 1.2: Start an instance of the CC-LINK IE TSN master stack.....	13
Figure 1.3: Sample implementation of the application callback	17
Figure 1.4: Configuration of a CANopen remote station	18
Figure 1.5: CANopen startup command table entry	18
Figure 1.6: Register a CANopen callback handler	18
Figure 2.1: Creating a new instance of the Remote Station stack	27
Figure 2.2: Start an instance of the Remote Station stack.....	28
Figure 2.3: Sample implementation of the application callback	30
Figure 2.4: Sample of CANopen Remote Station Setup	32
Figure 3.1: ICC Overview	34
Figure 3.2: CSP+ Input Section	35

List of Tables

Table 1-1: Directory structure of the Master SDK.....	7
Table 1-2: Configuration Functions of the CC-Link IE TSN protocol stack.....	9
Table 1-3: Runtime behavior functions of the CC-Link IE TSN Master stack.....	11
Table 1-4: Functions for Remote Station configuration	13
Table 1-5: Functions used during Run phase	14
Table 1-6: Arguments of the Application Callback Handler	14
Table 1-7: Handling of Application Callback IDs.....	17
Table 1-8: Members of callback data of CANopen data callback.....	19
Table 1-9: Confirmed Supported Hardware for the Master SDK	20
Table 1-10: Configuration Macros of the Conformance Test sample application	24
Table 2-1: Directory structure of the Remote Station SDK	25
Table 2-2: Configuration Functions of the Remote Station protocol stack.....	27
Table 2-3: Runtime behavior functions of the Remote Station stack	28
Table 2-4: Functions used during Run phase in the Remote Station SDK.....	29
Table 2-5: Arguments of the Application Callback Handler in the Remote Station SDK.....	29
Table 2-6: Handling of Application Callback IDs in the Remote Station SDK	30
Table 2-7: CANopen Data Access Functions.....	31
Table 2-8: Confirmed Supported Hardware of the Remote Station SDK	33

Changelog

Version	Changes
1.0	Initial Release
1.1	<p>Chapter 2.7</p> <ul style="list-style-type: none"> - Added functions goal_cclleTsnNmtUpload, goal_cclleTsnNmtDownload, goal_cclleTsnSlaveProcTypeRead <p>Chapter 2.8</p> <ul style="list-style-type: none"> - Added callbacks GOAL_CCL_CB_IP_ADDR_DUPL and GOAL_CCL_CB_READPROCTYPE_RES <p>Chapter 2.10</p> <ul style="list-style-type: none"> - Added callbacks GOAL_CCL_CO_CB_NMT_UPL_RES and GOAL_CCL_CO_CB_NMT_DNL_RES <p>Chapter 3.1.5</p> <ul style="list-style-type: none"> - Updated description for OpenIL 1.9 - Fixed misspellings <p>Chapter 4</p> <p>Added new application settings</p>
1.2	<p>Add Remote Station SDK Section</p> <p>Add ICC Section</p>

1 Master Station SDK

The CC-Link IE TSN Master SDK is used to implement a Management Master station or a Control Master Station in a device. The SDK uses GOAL, port's Industrial Communication Framework. The SDK has the following directory structure.

Path	Description
appl/goal_ccl_ie_tsn/01_master	Sample application for communication with port's Remote Station SDK
appl/goal_ccl_ie_tsn/03_master_ct	Sample application for Conformance Test
goal*	GOAL core, platform independent
plat	Platform specific files: architecture, board configuration, drivers
projects/goal_ccl_ie_tsn/01_master	Sample project for communication with port's Remote Station SDK
projects/goal_ccl_ie_tsn/03_master_ct	Sample project for Conformance Test
protos/ccl_ie_tsn	CC-Link IE TSN protocol stack
protos/goal_ts	(g)PTP protocol stack
protos/acd	ACD protocol stack
protos/slmp	SLMP protocol stack

Table 1-1: Directory structure of the Master SDK

1.1 Writing an Application for the Master SDK

1.1.1 Overview

An application for the Master SDK is a GOAL application. It consists of three functions that are called by GOAL: `appl_init`, `appl_setup`, `appl_loop`.

Additionally, the application can register a callback that is called by the CC-Link IE TSN Master stack to inform the application about events.

The function `appl_init` is used to register components in GOAL, e.g. the CC-Link IE TSN protocol stack.

The actual initialization of the application happens in `appl_setup`.

The application must include `goal_includes.h` and `goal_ccl_ie_tsn_master.h`.

1.1.2 Configuration of the CC-Link IE TSN protocol stack

The function `appl_setup` is called by GOAL during initialization. Within this function all functions listed in this chapter can be used to configure the behavior of the Management Master. All Functions must be called before calling `goal_cclleTsnNew`.

All functions return a status code indicating whether the operation succeeded or not.

Function	Description
<code>goal_cclleTsnCfgManagementPrioritySet</code>	Set the Management Priority of this master station
<code>goal_cclleTsnCfgNumGmRecordEntriesSet</code>	Set the maximum number of Grandmaster Record entries
<code>goal_cclleTsnCfgNumSlavesSet</code>	Set the maximum number of Remote Stations handled by this station
<code>goal_cclleTsnCfgNumS2sSubpayloads</code>	Set the maximum number of Subpayloads for direct communication between remote stations
<code>goal_cclleTsnCfgSlaveCheckIntervalSet</code>	Set the check interval for pending Remote Stations during RUN phase
<code>goal_cclleTsnCfgNodeTypeSet</code>	Set the node Type of this station (Management Master or Control Master)
<code>goal_cclleTsnCfgCertificationClassSet</code>	Set the Certification Class of this station
<code>goal_cclleTsnCfgDeviceVersionSet</code>	Set the Device Version of this station
<code>goal_cclleTsnCfgDeviceVendorCodeSet</code>	Set the Device Vendor Code of this station
<code>goal_cclleTsnCfgDeviceProductIdSet</code>	Set the Product Id of this station
<code>goal_cclleTsnCfgDeviceExModelCodeSet</code>	Set the Device Expansion Model Code of this station
<code>goal_cclleTsnCfgDeviceTypeIdSet</code>	Set the Device Type Id of this station
<code>goal_cclleTsnCfglogSyncIntSet</code>	Set the default logarithmic Sync Tx interval
<code>goal_cclleTsnCfglogAnnounceIntSet</code>	Set the default logarithmic Announce Tx interval
<code>goal_cclleTsnCfglogPDelayIntSet</code>	Set the default logarithmic PDelay Tx interval

goal_cclleTsnCfgPdelayResTimeSet	Set the time between Pdelay_Req and Pdelay_Resp_Follow_Up
goal_cclleTsnCfgDelaySetTimeSet	Set the time between Peer delay calculation and port role adjustment
goal_cclleTsnCfgAnnounceRelayTimeSet	Set the time for relaying Announce frames from Remote Station port to Master ports
goal_cclleTsnCfgNumTxSubpayloadEntrySet	Set the number of allowed Tx Subpayload information entries. The number defines how many Transmit Subpayloads can be handled by this station.
goal_cclleTsnCfgNumRxSubpayloadEntrySet	Set the number of allowed Rx Subpayload information entries. The number defines how many Receive Subpayloads can be handled by this station.
goal_cclleTsnCfgNumCycTxHandlersSet	Set the number of cyclic transmission handlers. The number defines how many cyclic connections can be established at the same time.
goal_cclleTsnCfgNumCycRxHandlersSet	Set the number of cyclic reception handlers. The number defines how many cyclic connections can be established at the same time.
goal_cclleTsnCfgNumSlmpServerHandlesSet	Define how many received SLMP requests can be processed in parallel
goal_cclleTsnCfgNumSlmpClientHandlesSet	Defines how many SLMP requests can be sent in parallel.
goal_cclleTsnCfgNumSlmpDivDataHandlesSet	Define how many fragmented SLMP messages can be received in parallel.
goal_cclleTsnCfgClearOnHoldEnable	Defines whether imported cyclic data is cleared or held if the the sender's application is stopped.
goal_cclleTsnCfgPtpPrio1Set	Overwrite the PTP prio1 value for this station.
goal_cclleTsnCfgLinkSpeedEnforce	Enforce a link speed for all ports

Table 1-2: Configuration Functions of the CC-Link IE TSN protocol stack

1.1.3 Creating a new instance of the CC-Link IE TSN protocol stack

After the stack has been configured the function *goal_cclleTsnNew* must be invoked to create a new instance of the protocol stack. It is also used to register a callback handler for processing events from the stack. The callback handler will be explained in detail in a later chapter.

```

GOAL_STATUS_T res;                               /* result */
static GOAL_CCL_HANDLE_T *pCcl = NULL;          /**< GOAL CCL handle */

/* create instance of CC-Link IE TSN stack */
res = goal_cclIeTsnNew(&pCcl, GOAL_CCL_INSTANCE_DEFAULT, appl_goalCclCb);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to instantiate CC-Link IE TSN stack");
    return res;
}

```

Figure 1.1: Creating a new instance of the CC-Link IE TSN master stack

This function creates a handle (in this example it is called pCcl) that must be used for all other function calls to reference the the stack instance.

1.1.4 Runtime behavior of the CC-Link IE TSN Master stack

The functions in this chapter influence the runtime behaviour of the CC-Link IE TSN Master Station. The functions must be called after *goal_cclleTsnNew* returned successfully and before *goal_cclleTsnStart* is called. These functions can be directly called within *appl_setup* or at a later point in time. These functions represent settings that usually come from an Engineering Tool. Therefore, the settings can be applied after receiving the current configuration from the tool.

Function	Description
<i>goal_cclleTsnDetectionAckEnforce</i>	Enforce DetectionAck transmission with every Detection frame
<i>goal_cclleTsnNetworkConfigSet</i>	set Network properties: - cycle time - CC-Link IE Field Coexistence - number of Generic PTP devices - TxProhibit time - timeslot number for cyclic communication - number of allowed consecutive cyclic errors
<i>goal_cclleTimeSlotAdd</i>	Add a time slot in ascending order from TSLT0 to TSLT7.
<i>goal_cclleTimeSlotEtherTypeAdd</i>	Assign an Ethertype to a timeslot in ascending order from TSLT1 to TSLT7.
<i>goal_cclleTimeSlotMacAddrAdd</i>	Assign a Destination MAC address to a specific timeslot.
<i>goal_cclleTimeSlotVlanAdd</i>	Assign a VLAN Tag to a specific timeslot.
<i>goal_cclleTsnPortFilterSet</i>	Set a port filter for each port. Use the <i>GOAL_CCL_PORT_FILTER_*</i> macros.
<i>goal_cclleTsnMulticastGroupAdd</i>	Register a Multicast Group for reception.
<i>goal_cclleTsnRxAddrOverlapCheckEnable</i>	Request Remote Stations to check their Rx Memory configurations for overlaps.
<i>goal_cclleTsnCycleCounterIgnoreEnable</i>	Instruct all Remote Stations to ignore the cycle counter in cyclic frames.
<i>goal_cclleTsnTimeoutsGet</i>	Get the timeout values of the Management Master state machine.
<i>goal_cclleTsnTimeoutsSet</i>	Set the timeout values of the Management Master state machine.

goal_cclleTsnTimeSyncSet	Set the time synchronization settings: <ul style="list-style-type: none"> - use 802.1AS or 1588v2 - domain number - logarithmic Sync interval - Sync Rx timeout factor - logarithmic Announce interval - announce Rx timeout - logarithmic Pdelay interval - DelayResp monitoring interval - delay mechanism (E2E or P2P) - number of tolerable Sync losses - synchronization tolerance
goal_cclleTsnNetworkNumberSet	Set the SLMP network number
goal_cclleTsnMasterIdSet	Set the Id of the master station
goal_cclleTsnCanOpenCallbackSet	Register a CANopen callback handler
goal_cclleTsnStationNumSet	Set the station number of the device.
goal_cclleTsnStationModeAdd	Register a Station mode for this device
goal_cclleTsnLinkDevAdd	Add a Link device to a Station Mode

Table 1-3: Runtime behavior functions of the CC-Link IE TSN Master stack

1.1.5 Remote Station Configuration

For each Remote Station station in the network a Remote Station handle must be added. Each Remote Station is represented by an Id. The functions must be called after *goal_cclleTsnNew* returned successfully and before *goal_cclleTsnStart* is called. These functions can be directly called within *appl_setup* or at a later point in time, i.e. after the expected Remote Station configuration has been received from the Engineering Tool.

Function	Description
goal_cclleTsnSlaveAdd	Add a Remote Station station with an IPv4 address and the ID of the Control Master. The function returns the Remote Station ID (used as internal reference).
goal_cclleTsnControlMasterAdd	Add a Control Master station with an IPv4 address and ist Master ID. The function returns the station handle ID (used as internal reference).
goal_cclleTsnSlaveStationModeSet	Set the expected station mode of a Remote Station.
goal_cclleTsnSlaveTsltMagnificationSet	Set the timeslot magnification value for a Remote Station.
goal_cclleTsnSlavePortFilterSet	Set the port filter for a Remote Station. Each array entry represents a port. Use the <code>GOAL_CCL_PORT_FILTER_*</code> macros.

goal_cclleTsnSlaveCyclicConfigSet	Set the cyclic configuration for a Remote Station: <ul style="list-style-type: none"> - if Remote Station is a reserved station - if Control data can be split in multiple frames - EMG groups Remote Station belongs to - GOF groups Remote Station belongs to - number of sub cycles - subCycle where frame is either sent or received - if cyclic frames must be sent from all ports
goal_cclleTsnSlaveInputAdd	This function is used to register an Input Link device for a Remote Station, i.e. a Subpayload transmitted by the Remote Station to the master. It requires the receive address in the master's memory, the transmitt address in the Remote Station's Link Device and the data length.
goal_cclleTsnSlaveOutputAdd	This function is used to register an Output Link device for a Remote Station, i.e. a Subpayload received by the Remote Station from the master. It requires the transmit address in the master's memory, the receive address in the Remote Station's Link Device and the data length.
goal_cclleTsnSlaveS2sTxSplAdd	Add a Tx Subpayload for direct communication to a Remote Station. The destination MAC address and IP address of the receiving Remote Station must be specified. Furthermore the Subcycle settings must be specified.
goal_cclleTsnSlaveS2sRxSplAdd	Add a Rx Subpayload for direct communication to a Remote Station.
goal_cclleTsnSlaveS2sRxSrcAdd	Add a Rx Source information item for direct communication to a Remote Station. This function sets the IP address of the sending Remote Station and the subcycle settings.
goal_cclleTsnSlaveRxMulticastGroupSet	Add a Remote Station to a Rx multicast group. The Remote Station's Output Subpayloads will be sent with other Subpayloads via Multicast frames.
goal_cclleTsnSlaveTxMulticastGroupSet	Add a Remote Station to a Tx multicast group. The Remote Station's Input Subpayloads will be received with other Subpayloads via Multicast frames.
goal_cclleSlaveTimeSlotMacAddrAdd	Assign a Destination MAC address to a timeslot for this Remote Station.
goal_cclleSlaveTimeSlotVlanAdd	Assign a VLAN to a timeslot for this Remote Station.
goal_cclleTsnSlaveUncontrolledMulticastGroupAdd	Add a Remote Station to a Multicast Group that is not controlled by this station, i.e. multicast group for Remote Station-to-Remote Station communication

goal_cclleTsnSlaveAppNetSyncEnable	Request the Remote Station to synchronize its application to the network cycle.
goal_cclleTsnSlaveCyclicStopEnable	Enable or disable the Cyclic Stop request for a Remote Station station.
goal_cclleTsnSlaveCanOpenConfigSet	Configure a Remote Station for CANopen communication. Register startup commands, and TPDO and RPDO config objects.

Table 1-4: Functions for Remote Station configuration

1.1.6 Starting the CC-Link IE TSN Master Stack

After finishing all settings of the runtime behavior and the expected Remote Station configuration the stack must be started by calling *goal_cclleTsnStart*.

```
res = goal_cclIeTsnStart(pCcl1);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to start stack");
}
```

Figure 1.2: Start an instance of the CC-LINK IE TSN master stack

If this function succeeds the stack CC-Link IE TSN Master stack has been started and tries to detect all configured Remote Stations. All functions described in previous chapters cannot be used anymore.

1.1.7 Functions used during Run phase

After successfully starting the stack. The application can use the following functions to access cyclic data and to influence the stack's behaviour.

Function	Description
goal_cclleTsnCyclicStopSet	Enable or Disable Cyclic Stop for the Master Station.
goal_cclleTsnInputGet	Read Input data from the Cyclic Memory Map. Only Input Link devices can be used. Each Link Device has its own memory map.
goal_cclleTsnOutputSet	Write Output data to the Cyclic Memory Map Only Output Link devices can be used. Each Link Device has its own memory map.
goal_cclleTsnAppStopSet	Enable or Disable Application Stop mode for this station.
goal_cclleTsnAppErrorStopSet	Enable or Disable Application Error Stop mode for this station.

goal_cclleTsnEmergencyStopExec	Issue an Emergency Stop due to internal error. Calling this function will cause all controlled devices in the network to shutdown. After calling this function the application is expected to halt the device and stop processing cyclic data.
goal_cclleTsnPowerSupplyErrorStopExec	Issue an Emergency Stop due to Power supply error. Calling this function will cause all controlled devices in the network to shutdown. After calling this function the application is expected to halt the device and stop processing cyclic data.
goal_cclleTsnOutputDevGet	Get data from an Output Link Device (data received via Master-to-Master communication).
goal_cclleTsnInputDevSet	Set data of an Input Link Device (data transmitted for Master-to-Master communication).
goal_cclleTsnSdoWrite	Start a SDO Write operation for a Remote Station.
goal_cclleTsnSdoRead	Start a SDO Read operation for a Remote Station.
goal_cclleTsnNmtUpload	Get the NMT state of a CANopen Remote Station.
goal_cclleTsnNmtDownload	Set the NMT state of a CANopen Remote Station.
goal_cclleTsnSlaveProcTypeRead	Request processor Type information from a Remote Station

Table 1-5: Functions used during Run phase

1.1.8 Application Callback

During initialization, the application can register a callback handler with the function *goal_cclleTsnNew*.

The callback handler uses the following arguments:

Argument data type	Description
GOAL_CCL_HANDLE_T *	CC-Link IE TSN stack instance reference
GOAL_CCL_CB_ID_T	callback ID indicating callback type
GOAL_CCL_CD_DATA_T *	callback data, actual meaning depends on callback ID

Table 1-6: Arguments of the Application Callback Handler

Some callback ID also evaluate the return value of the handler to decide how to proceed.

Callback ID	Description	Callback data	Return value
GOAL_CCL_CB_DETECTION_MISMATCH	mismatch in detected Remote Stations and	NULL	don't care

	configuration from engineering tool		
GOAL_CCL_CB_NETCFG_MISMATCH	mismatch in network configuration	pNetConfigMismatchId (Id of Remote Station with mismatching network configuration)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_MASTERCFG_MISMATCH	mismatch in configuration of a Control Master	NULL	don't care
GOAL_CCL_CB_ERROR_STATE	device entered Error state	NULL	don't care
GOAL_CCL_CB_UNKNOWN_SLAVE	detected a Remote Station that was not set by the engineering tool	pUnknownSlaveIpAddr (IP Address of unknown Remote Station)	don't care
GOAL_CCL_CB_SLAVE_UNDETECTED	Remote Station was not detected in the network	pUndetectedSlaveId (ID of Remote Station Handle)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_SLAVE_WRONG_IP_ADDR	Remote Station has wrong IP address (not in the same subnet)	pWrongIpAddrSlaveId (ID of Remote Station Handle)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_SLAVE_IP_ADDR_DUPL	Remote Station has a duplicate IP address	pDuplicateIpAddrSlaveId (ID of Remote Station Handle)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_SLMP_ERROR	SLMP error received from station	pSlmpErrorInfo->slaveId (ID of Remote Station Handle) pSlmpErrorInfo->cmd (SLMP command) pSlmpErrorInfo->subCmd (SLMP subcommand) pSlmpErrorInfo->endCode (end code indicating error)	don't care

GOAL_CCL_CB_CM_UNCONFIGURED	Control Master is not configured	pUnconfiguredMasterId (ID of Control Master handle)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_SLAVE_UNCONTROLLED	Remote Station not controlled by its Control Master	pUncontrolledSlaveId (ID of Remote Station Handle)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_WRONG_GRANDMASTER	Remote Station has a different Grandmaster	pWrongGrandmasterSlaveId (ID of Remote Station Handle)	don't care
GOAL_CCL_CB_RSV_TRANSIENT_DONE	reserved transient transmission done	NULL	don't care
GOAL_CCL_CB_RESERVED_STATION_ON	device entered Reserved Station mode	NULL	don't care
GOAL_CCL_CB_RESERVED_STATION_OFF	device left Reserved Station mode	NULL	don't care
GOAL_CCL_CB_CYCLIC_STOP_ON	cyclic communication stopped	NULL	don't care
GOAL_CCL_CB_CYCLIC_STOP_OFF	cyclic communication restarted	NULL	don't care
GOAL_CCL_CB_OWN_STATION_EMG_STOP	device received Emergency Stop request	pEmgGroup (EMG group causing stop)	don't care
GOAL_CCL_CB_CYCLIC_ERROR_ON	other station causes cyclic error	pCyclicErrIpAddr (IP address of Remote Station)	don't care
GOAL_CCL_CB_CYCLIC_ERROR_OFF	other station fixed cyclic error	pCyclicErrIpAddr (IP address of Remote Station)	don't care
GOAL_CCL_CB_OTHER_STATION_EMG_STOP	Emergency stop for an Emergency Group	pEmgGroup (EMG group that will be stopped)	don't care
GOAL_CCL_CB_OTHER_STATION_GOF_STOP	Emergency stop for a GOF Group	pGofGroup (GOF group that will be stopped)	don't care
GOAL_CCL_CB_SLAVE_INVALID_CFG	expected configuration of Remote Station is not valid	pInvalidCfgSlaveId (ID of Remote Station Handle)	GOAL_OK: continue operation other: go to Error state
GOAL_CCL_CB_SLAVE_INVALID_DATA_ON	received invalid control data from Remote Station	pInvalidCycDataSlaveId (ID of Remote Station Handle)	don't care

GOAL_CCL_CB_SLAVE_INVALID_DATA_OF F	control data from Remote Station is valid again	pInvalidCycDataSlaveId (ID of Remote Station Handle)	don't care
GOAL_CCL_CB_CYC_COM_ENABLED	station started to send and receive process data	NULL	don't care
GOAL_CCL_CB_CYC_COM_DISABLED	station stopped to send and receive process data	NULL	don't care
GOAL_CCL_CB_IP_ADDR_DUPL	station's IP address is also used by another station	NULL	don't care
GOAL_CCL_CB_READPROCTYPE_RES	received a ReadProcType response	pReadProcTypeRes (ReadProcType response data)	don't care

Table 1-7: Handling of Application Callback IDs

```

static GOAL_STATUS_T appl_goalCclCb(
    GOAL_CCL_HANDLE_T *pCclm,           /**< GOAL CCL handle */
    GOAL_CCL_CB_ID_T cbId,             /**< callback ID */
    GOAL_CCL_CD_DATA_T *pCbData       /**< callback data */
)
{
    GOAL_STATUS_T res = GOAL_OK;       /* result */

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

        case GOAL_CCL_CB_SLAVE_WRONG_IP_ADDR:
            goal_logInfo("slave 0x%04x has unexpected IP address",
                *(pCbData->pWrongIpAddrSlaveId));
            /* abort initialization */
            res = GOAL_ERROR;
            break;

        /* ... */
    }

    return res;
}

```

Figure 1.3: Sample implementation of the application callback

1.1.9 Configuration of CANopen Remote Stations

CANopen Remote Stations are configured with the same functions as Link Device Remote Stations. Additionally, there is the function *goal_cclleTsnSlaveCanOpenConfigSet* to set CANopen specific properties of the Remote Station.

```

static uint16_t slave2RpdoMapObj = 0x1601;      /**< Slave2: RPDO Mapping object */
static uint16_t slave2TpdoMapObj = 0x1A01;      /**< Slave2: TPDO Mapping object */
static GOAL_CCL_CO_STARTUP_T slave2StartUpCmds[] = {
    {GOAL_TRUE, 0x1C00, 1, sizeof(uint16_t), (uint8_t *) &slave2RpdoMapObj},
    {GOAL_TRUE, 0x1C01, 1, sizeof(uint16_t), (uint8_t *) &slave2TpdoMapObj},
};
static uint16_t slave2TpdoCfgTbl[] = {
    0x1C01,
};
static uint16_t slave2RpdoCfgTbl[] = {
    0x1C00,
};

res = goal_cclIeTsnSlaveCanOpenConfigSet(pCcl, slaveId,
                                         slave2StartUpCmds, ARRAY_ELEMENTS(slave2StartUpCmds),
                                         slave2TpdoCfgTbl, ARRAY_ELEMENTS(slave2TpdoCfgTbl),
                                         slave2RpdoCfgTbl, ARRAY_ELEMENTS(slave2RpdoCfgTbl));
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to configure CANopen properties of slave %u", slaveId);
}

```

Figure 1.4: Configuration of a CANopen remote station

An entry in the startup command table has the following fields:

```

typedef struct {
    GOAL_BOOL_T wrFlag;          /**< write or read access */
    uint16_t index;             /**< object index */
    uint16_t subIndex;          /**< object subindex */
    uint16_t dataLen;           /**< size of data to be written or read */
    uint8_t *pData;             /**< write data buffer */
} GOAL_CCL_CO_STARTUP_T;

```

Figure 1.5: CANopen startup command table entry

The other two tables contain lists of TPDO Config Objects and RPDO Config objects that must be enabled for cyclic communication.

In the RUN phase the application can use the functions `goal_cclIeTsnSdoWrite` and `goal_cclIeTsnSdoRead` to start SDO write or read operations.

1.1.10 CANopen Data Callback

The application can use the function `goal_cclIeTsnCanOpenCallbackSet` to register a callback handler for CANopen object access.

```

res = goal_cclIeTsnCanOpenCallbackSet(pCcl, appl_goalCclCanOpenCb);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to register CANopen callback");
}

```

Figure 1.6: Register a CANopen callback handler

The handler is called by the protocol stack every time a SDO Read or Write response was received, including those of the startup commands.

The callback handler uses the callback IDs GOAL_CCL_CO_CB_SDO_READ_RES or GOAL_CCL_CO_CB_SDO_WRITE_RES to indicate if the callback data refers to a read response or a write response.

The callback IDs GOAL_CCL_CO_CB_NMT_UPL_RES and GOAL_CCL_CO_CB_NMT_DNL_RES are used to indicate results of an NMT Upload or Download request.

Member	Data type	Description
slaveId	uint16_t	Remote Station id
endCode	uint16_t	SLMP end code (status of operation)
index	uint16_t	object index (only valid during startup or if endCode == 0x0000)
subIndex	uint8_t	object subindex (only valid during startup or if endCode == 0x0000)
dataLen	uint16_t	object data length (only valid for read access and if endCode == 0x0000)
pData	uint8_t *	object data (only valid for read access and if endCode == 0x0000) or current NMT state of responder (as a uint8_t variable)

Table 1-8: Members of callback data of CANopen data callback

1.2 Supported Platforms

Since the CC-Link IE TSN Protocol stack runs on GOAL, it can run on any platform supported by GOAL. However, to fulfill the requirements of a Class B device special hardware support is needed. The hardware must support timestamping of Ethernet frames as defined by IEEE 1588v2 or IEEE 802.1AS.

Additionally, it must support time aware queuing of Ethernet frames as defined by IEEE 802.1Qbv.

As development is constantly ongoing, the following table should not be seen as complete.

Confirmed Supported Hardware	Features
NXP LS1028A	Conformance Class B Hardware Qbv Support IEEE1588v2 and IEEE802.1AS 100Mbit/1000Mbit

Table 1-9: Confirmed Supported Hardware for the Master SDK

1.2.1 NXP LS1028A

The NXP LS1028A is a SoC that fulfills the hardware requirements for a Class B device. There is an evaluation board called LS1028ARDB.

1.2.1.1 Building the firmware

The LS1028ARDB uses OpenIL, a Linux distribution for industrial automation with Realtime support. The distribution can be built with Buildroot.

- git clone <https://github.com/openil/openil.git>
- cd openil
- git checkout OpenIL-v1.9-202009
- make nxp_ls1028ardb-64b_defconfig
- make 2>&1 | tee build.log

1.2.1.2 Flashing the firmware

Once the firmware has been built it must be copied to the SD-Card of the LS1028ARDB.

- insert SD-Card into Linux PC
 - card is listed as /dev/sdX [e.g. /dev/sdc]
 - see log via dmesg
- in directory openil:
 - sudo dd if=output/images/sdcard.img of=/dev/sdc bs=1024
 - **ATTENTION: choosing the wrong output device will overwrite sections of the PC's HDD/SDD causing data loss**

- insert SD-Card into LS1028ARD and start device

1.2.1.3 Debug Interface

A serial console is available on UART1.

Use the following parameters: 115,200 baud/s, 8 data bits, no parity, 1 stop bit.

1.2.1.4 Configuration of the LS1028ARDB

It is possible to configure the number of ports used for CC-Link IE TSN. By default each port of the integrated TSN Switch is independent, i.e. there is no forwarding between these ports.

The interfaces representing these ports are called swp0 to swp3. In order to enable forwarding between two or more ports the following script could be executed:

```
#!/bin/sh

# create a bridge device
ip link add name switch type bridge
ip link set switch up
# add 1st port
ip link set swp0 master switch
ip link set swp0 up
# add 2nd port
ip link set swp1 master switch
ip link set swp1 up
# uncomment to add 3rd port
#ip link set swp2 master switch
#ip link set swp2 up
# uncomment to add 4th port
#ip link set swp3 master switch
#ip link set swp3 up

# add a route for this interface (subnet address might need to adjusted)
ip route add 192.168.3.0/24 dev switch
```

This will create a new interface called “switch”.

By default, the OpenIL image runs netopeer2, a NETCONF server. This server interferes with the Realtime behaviour of the GOAL process. Therefore, the server must be removed from the initialization scripts:

```
rm /etc/init.d/S91netopeer2-server
```

1.2.1.5 Building the CC-Link IE TSN Master Application

The application can be built with aarch64-linux-gnu-gcc.

Navigate to the project folder of a sample application, e.g. “projects/goal_ccl_ie_tsn/03_master_ct/gcc”.

- Select the target platform (only required once)
 - make select
 - enter the number of "linux_nxp_ls1028a"
- build sample application
 - make

The binary is located at

"*projects/goal_ccl_ie_tsn/03_master_ct/gcc/linux_nxp_ls1028a/goal_linux_nxp_ls1028a.bin*".

The file must be copied to the LS1028ARDB, e.g. via scp.

On the LS1028ARDB:

- reduce kernel log messages to only critical ones
 - echo 1 > /proc/sys/kernel/printk
- disable memory overcommitting
 - echo 2 > /proc/sys/vm/overcommit_memory
- make application executable
 - chmod +x goal_linux_nxp_ls1028a.bin
- start application (adjust Ethernet port if necessary)
 - ./goal_linux_nxp_ls1028a.bin -i <IFACE>

Note:

<IFACE> is the ethernet interface that GOAL should use. This is either the bridge device "switch" or a standalone port, e.g. "swp0".

1.2.1.6 Start CC-Link IE TSN Master application automatically

In order to start the Master SDK automatically after boot up, a script must be created in the directory */etc/init.d*, e.g. S99GOAL.

The script should have the following content:

```
#!/bin/sh
#
# CC-Link IE TSN Master SDK
#
GOAL=/root/goal_linux_nxp_ls1028a.bin
start() {
    echo 1 > /proc/sys/kernel/printk
    echo 2 > /proc/sys/vm/overcommit_memory
    printf "Starting CC-Link IE TSN Master SDK: "
    ${GOAL} -i swp0 &
    echo "OK"
}
stop() {
    printf "Stopping CC-Link IE TSN Master SDK: "
    killall $(basename ${GOAL})
    echo "OK"
}
```

```
}
restart() {
    stop
    start
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart|reload)
        restart
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
esac
```

Please make sure that the script is executable:
chmod +x /etc/init.d/S99GOAL

1.3 Conformance Test

In order to perform the conformance test, several configurations must be tested.

Configuration of the Master Station the network and the expected Remote Stations is done by API functions as described in chapter 2.

There is a sample application that contains the required configurations. It can be found in *appl/goal_ccl_ie_tsn/03_master_ct*.

There are configuration macros to enable or disable features. If a new configuration is required, the macros must be set to the appropriate values. The application must be recompiled (see chapter 3.1.5) and copied to the LS1028ARDB (see chapter 3.1.6).

Configuration Macro	Description
GOAL_APPL_SLAVE0_ENABLED	1: enable Remote Station 0 0: disable Remote Station 0 (NZ2GN2B1-32DTE, 192.168.3.1)
GOAL_APPL_SLAVE1_ENABLED	1: enable Remote Station 1 0: disable Remote Station 1 (RJ71GN11-T2, 192.168.3.2)
GOAL_APPL_SLAVE2_ENABLED	1: enable Remote Station 0 0: disable Remote Station 0 (MR-J5-10G, 192.168.3.10)
GOAL_APPL_SLAVE2_ALT_MAPPING	1: use an alternate PDO mapping for Remote Station 2 0: use default PDO mapping for Remote Station 2
GOAL_APPL_CO_OBJ_TEST	1: execute CANopen object test for Remote Station 2 0: do not execute test
GOAL_APPL_SLMP_TEST	1: execute SLMP Client test 0: do not execute test
GOAL_APPL_TIMESYNC_1588	1: use IEEE 1588v2 for time synchronization 0: use IEEE 802.1AS for time synchronization
GOAL_APPL_MULTICAST	1: use multicast frames for cyclic communication 0: use unicast frames for cyclic communication
GOAL_APPL_TIME_SYNC_SLAVE	1: device is not the Grandmaster 0: device is the Grandmaster
GOAL_APPL_LINKSPEED_100	1: enforce a Link speed of 100 Mbit/s 0: enforce a Link speed of 1000 Mbit/s

Table 1-10: Configuration Macros of the Conformance Test sample application

2 Remote Station SDK

The CC-Link IE TSN Remote Station SDK is used to implement a Remote Station in a device. The SDK uses GOAL, port's Industrial Communication Framework.

The SDK has the following directory structure.

Path	Description
appl/goal_ccl_ie_tsn/02_slave	Sample application for communication with a Master Station
goal*	GOAL core, platform independent
plat	Platform specific files: architecture, board configuration, drivers
projects/goal_ccl_ie_tsn/02_slave	Sample project
protos/ccl_ie_tsn	CC-Link IE TSN protocol stack
protos/goal_ts	(g)PTP protocol stack
protos/slmp	SLMP protocol stack
protos/acd	ACD protocol stack
protos/802_1Qbv	Goal Qbv connector for software Qbv
ext/802_1Qbv	software Qbv implementation

Table 2-1: Directory structure of the Remote Station SDK

2.1 Writing an Application for the Remote Station SDK

2.1.1 Overview

An application for the Remote Station SDK is a GOAL application. It consists of three functions that are called by GOAL: `appl_init`, `appl_setup`, `appl_loop`.

Additionally, the application can register a callback that is called by the CC-Link IE TSN Remote Station stack to inform the application about events.

The function `appl_init` is used to register components in GOAL, e.g. the CC-Link IE TSN protocol stack. The actual initialization of the application happens in `appl_setup`.

The application must include `goal_includes.h` and `goal_ccl_ie_tsn.h`.

2.1.2 Configuration of the CC-Link IE TSN protocol stack

The function `appl_setup` is called by GOAL during initialization. Within this function all functions listed in this chapter can be used to configure the behavior of the Remote Station. All Functions must be called before calling `goal_cclIeTsnNew`.

All functions return a status code indicating whether the operation succeeded or not.

Function	Description
<code>goal_cclIeTsnCfgNodeTypeSet</code>	Set the node Type of this station (Remote Station)
<code>goal_cclIeTsnCfgCertificationClassSet</code>	Set the Certification Class of this station
<code>goal_cclIeTsnCfgDeviceVersionSet</code>	Set the Device Version of this station
<code>goal_cclIeTsnCfgDeviceVendorCodeSet</code>	Set the Device Vendor Code of this station
<code>goal_cclIeTsnCfgDeviceProductIdSet</code>	Set the Product Id of this station
<code>goal_cclIeTsnCfgDeviceExModelCodeSet</code>	Set the Device Expansion Model Code of this station
<code>goal_cclIeTsnCfgDeviceTypeIdSet</code>	Set the Device Type Id of this station
<code>goal_cclIeTsnCfglogSyncIntSet</code>	Set the default logarithmic Sync Tx interval
<code>goal_cclIeTsnCfglogAnnounceIntSet</code>	Set the default logarithmic Announce Tx interval
<code>goal_cclIeTsnCfglogPDelayIntSet</code>	Set the default logarithmic PDelay Tx interval
<code>goal_cclIeTsnCfgPdelayResTimeSet</code>	Set the time between Pdelay_Req and Pdelay_Resp_Follow_Up
<code>goal_cclIeTsnCfgDelaySetTimeSet</code>	Set the time between Peer delay calculation and port role adjustment
<code>goal_cclIeTsnCfgAnounceRelayTimeSet</code>	Set the time for relaying Announce frames from Remote Station port to Remote Station ports
<code>goal_cclIeTsnCfgNumTxSubpayloadEntriesSet</code>	Set the number of allowed Tx Subpayload information entries. The number defines how many Transmit Subpayloads can be handled by this station.

goal_cclIeTsnCfgNumRxSubpayloadEntriesSet	Set the number of allowed Rx Subpayload information entries. The number defines how many Receive Subpayloads can be handled by this station.
goal_cclIeTsnCfgNumCycTxHandlersSet	Set the number of cyclic transmission handlers. The number defines how many cyclic connections can be established at the same time.
goal_cclIeTsnCfgNumCycRxHandlersSet	Set the number of cyclic reception handlers. The number defines how many cyclic connections can be established at the same time.
goal_cclIeTsnCfgNumSlmpServerHandlesSet	Define how many received SLMP requests can be processed in parallel
goal_cclIeTsnCfgNumSlmpClientHandlesSet	Defines how many SLMP requests can be sent in parallel.
goal_cclIeTsnCfgNumSlmpDivDataHandlesSet	Define how many fragmented SLMP messages can be received in parallel.
goal_cclIeTsnCfgClearOnHoldEnable	Defines whether imported cyclic data is cleared or held if the the sender's application is stopped.
goal_cclIeTsnCfgPtpPrio1Set	Overwrite the PTP prio1 value for this station.
goal_cclIeTsnCfgLinkSpeedEnforce	Enforce a link speed for all ports

Table 2-2: Configuration Functions of the Remote Station protocol stack

2.1.3 Creating a new instance of the CC-Link IE TSN protocol stack

After the stack has been configured the function *goal_cclIeTsnNew* must be invoked to create a new instance of the protocol stack. It is also used to register a callback handler for processing events from the stack. The callback handler will be explained in detail in a later chapter.

```

GOAL_STATUS_T res;                               /* result */
static GOAL_CCL_HANDLE_T *pCcl = NULL;          /**< GOAL CCL handle */

/* create instance of CC-Link IE TSN stack */
res = goal_cclIeTsnNew(&pCcl, GOAL_CCL_INSTANCE_DEFAULT, appl_goalCclCb);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to instantiate CC-Link IE TSN stack");
    return res;
}

```

Figure 2.1: Creating a new instance of the Remote Station stack

This function creates a handle (in this example it is called pCcl) that must be used for all other function calls to reference the stack instance.

2.1.4 Runtime behavior of the CC-Link IE TSN Remote Station stack

The functions in this chapter influence the runtime behaviour of the CC-Link IE TSN Remote Station. The functions must be called after *goal_cclIeTsnNew* returned successfully and before *goal_cclIeTsnStart* is called. These functions can be directly called within *appl_setup* or at a later point in time.

Function	Description
goal_cclleTsnCanOpenCallbackSet	Register a CANopen callback handler
goal_cclleTsnStationNumSet	Set the station number of the device.
goal_cclleTsnStationModeAdd	Register a Station mode for this device
goal_cclleTsnLinkDevAdd	Add a Link device to a Station Mode

Table 2-3: Runtime behavior functions of the Remote Station stack

2.1.5 Starting the CC-Link IE TSN Remote Station Stack

After finishing all settings of the runtime behavior and the expected Remote Station configuration the stack must be started by calling *goal_cclleTsnStart*.

```
res = goal_cclleTsnStart(pCcl);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to start stack");
}
```

Figure 2.2: Start an instance of the Remote Station stack

If this function succeeds the stack CC-Link IE TSN Remote Station stack has been started and will now wait for a configuration by a master device. All functions described in previous chapters cannot be used anymore.

2.1.6 Functions used during Run phase

After successfully starting the stack. The application can use the following functions to access cyclic data and to influence the stack's behaviour.

Function	Description
goal_cclleTsnCyclicStopSet	Enable or Disable Cyclic Stop for the Remote Station.
goal_cclleTsnInputGet	Read Input data from the Cyclic Memory Map. Only Input Link devices can be used. Each Link Device has its own memory map.
goal_cclleTsnOutputSet	Write Output data to the Cyclic Memory Map Only Output Link devices can be used. Each Link Device has its own memory map.
goal_cclleTsnAppStopSet	Enable or Disable Application Stop mode for this station.
goal_cclleTsnAppErrorStopSet	Enable or Disable Application Error Stop mode for this station.
goal_cclleTsnEmergencyStopExec	Issue an Emergency Stop due to internal error. Calling this function will cause all controlled devices in the network to shutdown. After calling this function the application is expected to halt the device and stop processing cyclic data.

goal_cclleTsnPowerSupplyErrorStopExec	Issue an Emergency Stop due to Power supply error. Calling this function will cause all controlled devices in the network to shutdown. After calling this function the application is expected to halt the device and stop processing cyclic data.
goal_cclleTsnOutputDevGet	Get data from an Output Link Device (data received via Master-to-Master communication).
goal_cclleTsnInputDevSet	Set data of an Input Link Device (data transmitted for Master-to-Master communication).
goal_cclleTsnNmtUpload	Get the NMT state of a CANopen Remote Station.
goal_cclleTsnSlaveProcTypeRead	Request processor Type information from a Remote Station

Table 2-4: Functions used during Run phase in the Remote Station SDK

2.1.7 Application Callback

During initialization, the application can register a callback handler with the function *goal_cclleTsnNew*.

The callback handler uses the following arguments:

Argument data type	Description
GOAL_CCL_HANDLE_T *	CC-Link IE TSN stack instance reference
GOAL_CCL_CB_ID_T	callback ID indicating callback type
GOAL_CCL_CD_DATA_T *	callback data, actual meaning depends on callback ID

Table 2-5: Arguments of the Application Callback Handler in the Remote Station SDK

Some callback ID also evaluate the return value of the handler to decide how to proceed.

Callback ID	Description	Callback data	Return value
GOAL_CCL_CB_ERROR_STATE	device entered Error state	NULL	don't care
GOAL_CCL_CB_RSV_TRANSIENT_DONE	reserved transient transmission done	NULL	don't care
GOAL_CCL_CB_RESERVED_STATION_ON	device entered Reserved Station mode	NULL	don't care
GOAL_CCL_CB_RESERVED_STATION_OFF	device left Reserved Station mode	NULL	don't care
GOAL_CCL_CB_CYCLIC_STOP_ON	cyclic communication stopped	NULL	don't care
GOAL_CCL_CB_CYCLIC_STOP_OFF	cyclic communication restarted	NULL	don't care
GOAL_CCL_CB_OWN_STATION_EMG_STOP	device received Emergency Stop request	pEmgGroup (EMG group causing stop)	don't care

GOAL_CCL_CB_CYCLIC_ERROR_ON	other station causes cyclic error	pCyclicErrIpAddr (IP address of Remote Station)	don't care
GOAL_CCL_CB_CYCLIC_ERROR_OFF	other station fixed cyclic error	pCyclicErrIpAddr (IP address of Remote Station)	don't care
GOAL_CCL_CB_CYC_COM_ENABLED	station started to send and receive process data	NULL	don't care
GOAL_CCL_CB_CYC_COM_DISABLED	station stopped to send and receive process data	NULL	don't care
GOAL_CCL_CB_IP_ADDR_DUPL	station's IP address is also used by another station	NULL	don't care
GOAL_CCL_CB_STATION_MODE_CHANGE	local station changed station mode	NULL	don't care

Table 2-6: Handling of Application Callback IDs in the Remote Station SDK

```

static GOAL_STATUS_T appl_goalCclCb(
    GOAL_CCL_HANDLE_T *pCclm,           /**< GOAL CCL handle */
    GOAL_CCL_CB_ID_T cbId,             /**< callback ID */
    GOAL_CCL_CD_DATA_T *pCbData       /**< callback data */
)
{
    GOAL_STATUS_T res = GOAL_OK;       /* result */

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

        case GOAL_CCL_CB_SLAVE_WRONG_IP_ADDR:
            goal_logInfo("slave 0x%04x has unexpected IP address",
                *(pCbData->pWrongIpAddrSlaveId));
            /* abort initialization */
            res = GOAL_ERROR;
            break;

        /* ... */
    }

    return res;
}

```

Figure 2.3: Sample implementation of the application callback

2.1.8 Data Access with CANopen

The Remote Station SDK can be extended with a CANopen package which provides an additional approach to configure the data transfer, using methods known originally adapted from the CANopen specification. Specifically, this uses the concepts of RPDO and TPDO configuration and PDO or SDO data transfer.

To set up this operation mode, the station mode needs to be marked as CANopen. Instead of Word link devices, CANopen link devices must be initialized. Additionally, this mode requires an object

dictionary and a CANopen configuration file. It is recommended to use the port ICC tool to interactively create them. (See chapter 3 in this document.)

When accessing the CANopen link devices, the general access function will not work. Instead, the following functions will address CANopen objects by index and subindex directly:

Function	Description
goal_cclleTsnInputDevSetCANopen	Write to object in object dictionary
goal_cclleTsnInputDevGetCANopen	Read from object in object dictionary

Table 2-7: CANopen Data Access Functions

```

/* set the station mode */
res = goal_cclIeTsnStationModeAdd(pCcl, GOAL_APPL_STATION_MODE_CANOPEN);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to set station mode");
    return res;
}

/* this is a CANopen station mode */
res = goal_cclIeTsnStationModeSetCANopen(pCcl, GOAL_APPL_STATION_MODE_CANOPEN);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to set station mode to CANopen mode");
    return res;
}

/* Statusword communication linkdevice */
res = goal_cclIeTsnLinkDevAdd(pCcl, GOAL_APPL_STATION_MODE_CANOPEN,
GOAL_CCL_LINK_DEV_STSW, 0x60, 8);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to set local linkDev");
    return res;
}

/* CANopen In -- CANopen uses a virtual address */
res = goal_cclIeTsnLinkDevAdd(pCcl, GOAL_APPL_STATION_MODE_CANOPEN,
GOAL_CCL_LINK_DEV_CANOPEN_IN, 0x00004000, 1000);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to set local linkDev");
    return res;
}

/* CANopen Out -- CANopen uses a virtual address */
res = goal_cclIeTsnLinkDevAdd(pCcl, GOAL_APPL_STATION_MODE_CANOPEN,
GOAL_CCL_LINK_DEV_CANOPEN_OUT, 0x00008000, 1000);
if (GOAL_RES_ERR(res)) {
    goal_logErr("Failed to set local linkDev");
    return res;
}

```

Figure 2.4: Sample of CANopen Remote Station Setup

2.2 Supported Platforms

Since the CC-Link IE TSN Protocol stack runs on GOAL, it can run on any platform supported by GOAL.

In the less strict Conformance Class A, the time sensitive sending of ethernet frames can be supported by the included SoftwareQbv package. The hardware only has to support timestamping of the frames.

To fulfill the requirements of a Conformance Class B device, special hardware support is needed. The hardware must support timestamping of Ethernet frames as defined by IEEE 1588v2 or IEEE

802.1AS. Additionally, it must support time aware queuing of Ethernet frames as defined by IEEE 802.1Qbv.

As development is constantly ongoing, the following table should not be seen as complete.

Confirmed Supported Hardware	Features
STM32F429ZI Nucleo Discovery Board	Conformance Class A Software Qbv IEEE1588v2 and IEEE802.1AS 100Mbit

Table 2-8: Confirmed Supported Hardware of the Remote Station SDK

2.3 Conformance Test

To perform the conformance test, several configurations must be tested. All of them were tested with the example application *appl/goal_ccl_ie_tsn/02_slave*. The respective master device or PLC is responsible for setting up the remote station for a specific test.

3 ICC support

ICC stands for Industrial Communication Creator; it is the unified tool by *port* for creating projects of any of the supported protocols. For the CC Link IE TSN stack, it helps creating a project configuration and the CSP+ device description file. For the Remote Station SDK, it is also able to create the local object dictionary for the CANopen support.

Note: It is possible (but not recommended) to create project configurations manually. For the CSP+ files, other tools may be used either on their own or together with the ICC. Creating an object dictionary without the ICC is not recommended.

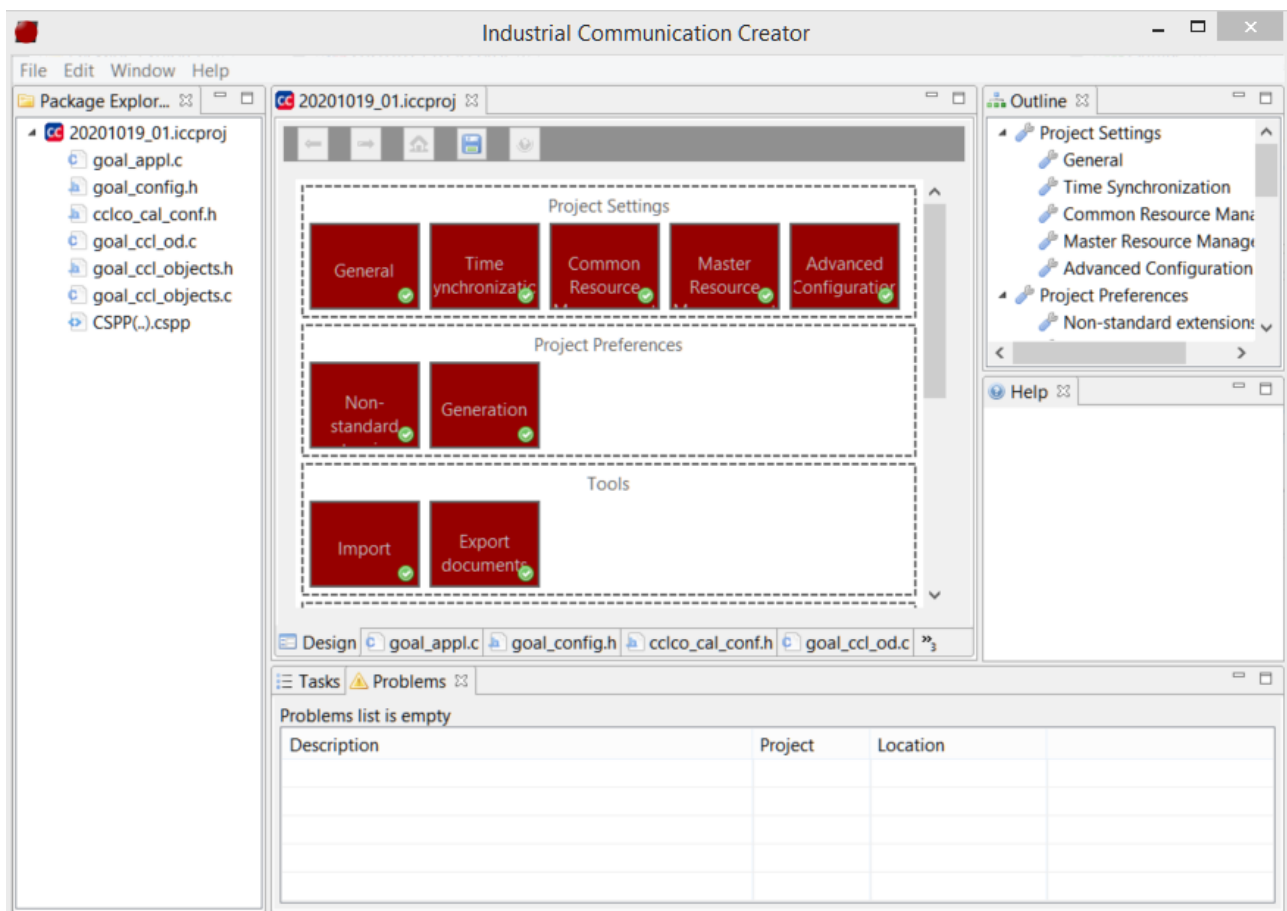


Figure 3.1: ICC Overview

Create a new project by filling in all the needed information. The following files will be created and should be added to the application:

- Goal_appl.c – an example application, usable as a base for development
- Goal_config.h – configuration file for the GOAL runtime
- Cclco_cal_conf.h – configuration file for the CANopen support (optional)
- Goal_ccl_od.c/.h – object dictionary for CANopen support (optional)
- Goal_ccl_objects.c/.h – object definitions for CANopen support (optional)
- CSPP.cspp – device description file

- ✦ CSP+ File description
 - ✦ Section 1 FileSection
 - ✦ Part 1 FileInfo
 - ✦ Element 1 CreateDate
 - ✦ Element 2 CreateTime
 - ✦ Element 3 ModDate
 - ✦ Element 4 ModTime
 - ✦ Element 5 Language
 - ✦ Element 6 CCLinkFamilyProfileVersion
 - ✦ Element 7 FileVersion
 - ✦ Section 2 DeviceSection
 - ✦ Part 1 DeviceInfo
 - ✦ Section 3 BlockSection
 - ✦ Part 1 BlockInfo
 - ✦ Element 1 VendorName
 - ✦ Element 2 VendorCode
 - ✦ Element 3 Version
 - ✦ Section 4 CommIfSection0
 - ✦ Part 1 CommIfInfo
 - ✦ Element 1 VendorName
 - ✦ Element 2 VendorCode
 - ✦ Element 3 CommIfTypeID
 - ✦ Element 4 Version
 - ✦ Element 5 StationMode
 - ✦ Element 6 StationModeName
 - ✦ Element 7 LocalFunction
 - ✦ Element 8 AutoSettingHeaderType
 - ✦ Element 9 AutoSettingType
 - ✦ Element 10 ModelCode
 - ✦ Element 11 ModelCodeEx
 - ✦ Element 12 DevModel

Figure 3.2: CSP+ Input Section

For further information check the manual of the ICC tool.