# Table of contents

1. **Introduction** .................................................................................................................. 5

2. **DALI** .................................................................................................................................. 6
   - 2.1 IEC 62386 ...................................................................................................................... 6
   - 2.2 Communication ............................................................................................................... 7
   - 2.3 Priorities ......................................................................................................................... 8
   - 2.4 Memory banks ................................................................................................................ 9

3. **Programming** .................................................................................................................. 12
   - 3.1 POUs ................................................................................................................................ 12
     - 3.1.1 Applications .............................................................................................................. 12
     - 3.1.2 Commands ............................................................................................................... 36
     - 3.1.3 Communication ....................................................................................................... 341
     - 3.1.4 Devices .................................................................................................................... 346
     - 3.1.5 Events ...................................................................................................................... 543
   - 3.2 DUTs ................................................................................................................................ 546
     - 3.2.1 Structures ................................................................................................................ 546
     - 3.2.2 Enumerations .......................................................................................................... 547
   - 3.3 GVLs .............................................................................................................................. 555
     - 3.3.1 Constants ................................................................................................................ 555
     - 3.3.2 Parameter ................................................................................................................. 556
   - 3.4 Integration into TwinCAT .............................................................................................. 557
     - 3.4.1 KL6821 with CX5120 ............................................................................................ 557
   - 3.5 Runtime messages ......................................................................................................... 561

4. **Appendix** ....................................................................................................................... 566
   - 4.1 Commissioning and diagnosis ....................................................................................... 566
     - 4.1.1 KS2000 .................................................................................................................... 566
     - 4.1.2 PLC HMI .................................................................................................................. 567
     - 4.1.3 Event Logger ............................................................................................................ 570
     - 4.1.4 Communication Monitor ......................................................................................... 572
     - 4.1.5 Error evaluation ....................................................................................................... 574
   - 4.2 Tc2_DALI and Tc3_DALI in the same project ............................................................... 577
   - 4.3 Creating an application function block .......................................................................... 579
1 Introduction

The user of this library requires basic knowledge of the following:

- TwinCAT XAE
- Design and characteristics of Beckhoff IPCs and their Bus Terminal system
- DALI technology and mode of operation
- Relevant safety regulations for building technical equipment

This software library is intended for building automation system partners of Beckhoff Automation GmbH & Co. KG. The system partners operate in the field of building automation and are concerned with the installation, commissioning, expansion, maintenance and service of measurement, control and regulating systems for the technical equipment of buildings.

The Tc3_DALI library is usable on all hardware platforms that support TwinCAT 3.1.4022.20 or higher.
DALI

DALI (Digital Addressable Lighting Interface) is a definition for the standardization of digital interfaces between control gears (lamps) and control devices (sensors). The standard (IEC 62386) allows the manufacturers of lighting components to implement complex lighting tasks easily and conveniently.

The KL6811 (DALI/DSI master) and KL6821 (DALI2 master) Bus Terminals are integrated into the Bus Terminal system as normal Bus Terminals and are therefore fieldbus-independent. The DALI data is forwarded to the DALI devices via the respective Bus Coupler. Bus controllers also offer the option of running PLC programs locally in IEC 61131-3.

2.1 IEC 62386

DALI is specified in the IEC 62386 standard and offers advantages such as flexibility, simplicity, user friendliness and robustness. IEC 62386 has been revised several times and was extended considerably in November 2014 with the publication of the second revision. While in the first revision only control gears (lamps) were considered, from the second revision onwards control devices (sensors) are also included. These are described in the respective section of IEC 62386:

<table>
<thead>
<tr>
<th>IEC 62386-101</th>
<th>General system properties such as cabling, feed-in and telegram structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC 62386-102</td>
<td>General properties of the control gears</td>
</tr>
<tr>
<td>IEC 62386-201</td>
<td>Fluorescent lamps (device type 0)</td>
</tr>
<tr>
<td>IEC 62386-202</td>
<td>Emergency lighting (device type 1)</td>
</tr>
<tr>
<td>IEC 62386-203</td>
<td>Discharge lamps (device type 2)</td>
</tr>
<tr>
<td>IEC 62386-207</td>
<td>LED modules (device type 6)</td>
</tr>
<tr>
<td>IEC 62386-103</td>
<td>General properties of the control devices</td>
</tr>
<tr>
<td>IEC 62386-301</td>
<td>Push-buttons</td>
</tr>
<tr>
<td>IEC 62386-302</td>
<td>Input devices, absolute input devices</td>
</tr>
<tr>
<td>IEC 62386-303</td>
<td>Occupancy sensor</td>
</tr>
<tr>
<td>IEC 62386-304</td>
<td>Brightness sensor</td>
</tr>
</tbody>
</table>

The IEC 62386-101, IEC 62386-102 and IEC 62386-103 standards describe general properties, while the IEC 62386-2xx and IEC 62386-3xx standards specify the individual device types. IEC 62386-103 and IEC 62386-3xx were included in Revision 2 of the DALI standard.
Up to 64 control gears and up to 64 control devices can be connected to the KL6821 per DALI line. The KL6821 represents the DALI controller. One such device exists for each DALI line. Up to 64 control gears, which have to be DALI/DSI devices, can be connected to the KL6811. Any number of DALI lines (KL6821 and KL6811) can be operated with a single TwinCAT controller.

### 2.2 Communication

With regard to the communication, a distinction is made between three telegram types:

- 16-bit query, configuration and control telegram.
- 24-bit query, configuration and control telegram.
- 24-bit event telegram.

#### 16-bit telegrams

16-bit telegrams are always sent from a DALI controller to a DALI control gear. They are used for configuring the devices, querying parameters or sending control commands. For certain DALI commands the DALI control gear sends an 8-bit response. DALI control gears only send an 8-bit telegram when requested.

In the DALI library these commands are provided in the form of PLC function blocks with the prefix FB_DALI102, e.g. FB_DALI102QueryActualLevel.

#### 24-bit telegrams

24-bit telegrams are always sent from a DALI controller to a DALI control device. They are used for configuring the devices, querying parameters or sending control commands. For certain DALI commands the DALI input device sends an 8-bit response.

In the DALI library these commands are provided in the form of PLC function blocks with the prefix FB_DALI103, e.g. FB_DALI103QueryOperatingMode.

#### 24-bit events

DALI input devices are able to send events. They are always evaluated by the DALI controller and have a length of 24 bits.

Individual events can be filtered out and further processed with the function blocks FB_DALIGetInputNotification and FB_DALIGetPowerCycleNotification.

Further information on DALI can be found on the homepage of the Digital Illumination Interface Alliance (https://www.dali-alliance.org) and in the IEC 62386 standard.
The KL6811 only supports the first revision of the DALI standard. It is not possible to operate control devices with the KL6811.

2.3 Priorities

Since DALI-2 is a multi-master bus system, simultaneous bus access of several DALI masters is controlled based on priorities. According to IEC 62386, all DALI-2 devices that can initiate sending of a DALI command (controllers) or sending of an event (input devices) on the DALI bus are referred to as masters. Under DALI-2, these are also referred to as control devices and are described in more detail in IEC 62386-103.

All DALI-2 devices of a DALI line must share the same data line. To avoid collisions during sending, the sending device checks whether the DALI bus has already been assigned. Sending takes place after a certain waiting time (settling time), once the DALI bus is free. For high-priority DALI commands the bus access takes place after a short waiting time, for low-priority commands the waiting time is longer. In other words, high-priority DALI commands are given preference over low-priority DALI commands.

DALI control gears are defined in IEC 62386-102. They are not capable of sending DALI commands or events independently. DALI control gears may only return the 8-bit backward frame to forward frames sent by a DALI controller (see also Communication). Since a DALI controller waits for the backward frame, the 8-bit backward frame has the shortest waiting time. This waiting time is shorter than for DALI commands with the highest priority. This means that DALI forward frames can be processed without interference from other DALI commands.

The priorities used by a DALI controller for sending the DALI commands are referred to as command priorities and are mapped by the data type `E_DALICommandPriority`. Command priorities can have 5 different values:

- **Low**: DALI-2 priority 5
- **Middle low**: DALI-2 priority 4
- **Middle**: DALI-2 priority 3
- **Middle high**: DALI-2 priority 2
- **High**: DALI-2 priority 1

Most function blocks referred to in chapter Part 102 (control gears) have the input `eCommandPriority`. This input is used to specify the priority with which the DALI commands are to be sent via the KL6821.

Events also have a priority (event priority), which is represented by the data type `E_DALIEventPriority`. Event priorities can have 4 values in the range **Low** (DALI-2 priority 5) to **Middle high** (DALI-2 priority 2). The event priority is written as a parameter (see instance variable Instance variables) into the respective instances of the DALI devices.

Priority **High** (DALI-2 priority 1) is only allowed for DALI-2 commands and cannot be used for events.

Tc3_DALI uses the following default values for the priorities:

<table>
<thead>
<tr>
<th>E_DALICommandPriority/E_DALIEventPriority</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>-</td>
</tr>
<tr>
<td>MiddleLow</td>
<td>Light sensor events (Part 304). All other DALI commands.</td>
</tr>
<tr>
<td>Middle</td>
<td>Events of push buttons (Part 301), absolute input devices (Part 302) and occupancy sensors (Part 303).</td>
</tr>
<tr>
<td>MiddleHigh</td>
<td>DALI commands for writing parameters and for addressing DALI devices.</td>
</tr>
<tr>
<td>High</td>
<td>DALI commands for transactions (from the second DALI command).</td>
</tr>
</tbody>
</table>
When selecting priorities, care should always be taken to ensure that time-critical events that are important for switching the lighting have a higher priority than the DALI commands themselves. Non-system-critical DALI commands, such as the cyclic querying of states for the display in a visualization, should be sent with a lower priority.

Priorities for DALI commands ([E_DALICommandPriority](#)) are supported from Tc3_DALI V3.5.0.0. The KL6821 must contain firmware BD or later. Older firmware versions always send DALI commands with the priority *High*. Neither the KL6811 nor the Tc2_DALI library support priorities for DALI commands.

Priorities are always important in situations where input devices send events and DALI commands are sent in parallel via the KL6821. If no input devices are used, the priority of the DALI commands is of secondary importance, since in this case there is only one DALI master (the KL6821).

### 2.4 Memory banks

Memory banks are freely accessible memory areas in which device-specific information and properties can be stored. Part of the memory banks is marked as read-only and contains further, vendor-specific information. The contents of the memory banks can be read with [FB_DALI10xReadMemoryLocation](#) (see [FB_DALI102ReadMemoryLocation](#) and [FB_DALI103ReadMemoryLocation](#)) and, if enabled, written to with [FB_DALI10xWriteMemoryLocationNoReply](#) (see [FB_DALI102WriteMemoryLocationNoReply](#) and [FB_DALI103WriteMemoryLocationNoReply](#)).

Part of the memory banks can be provided with write protection.

A DALI device can support a maximum of 256 memory banks, each with up to 255 bytes, with memory banks 200 to 255 being reserved. In the current standard, memory bank 0 and a large part of memory bank 1 are predefined.

**Structure of memory bank 0:**

Memory bank 0 contains information on the DALI control gear or DALI control device. It exists inside all of these devices.
<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Default values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Address of the last memory area inside the memory bank that can be accessed.</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x01</td>
<td>Reserved, not implemented</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>Number of the last memory bank that can be accessed.</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x03</td>
<td>GTIN byte0 (MSB)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x04</td>
<td>GTIN byte1</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x05</td>
<td>GTIN byte2</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x06</td>
<td>GTIN byte3</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x07</td>
<td>GTIN byte4</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x08</td>
<td>GTIN byte5 (LSB)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x09</td>
<td>Firmware Version (major)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x0A</td>
<td>Firmware Version (minor)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x0B</td>
<td>Identification number byte 0 (MSB)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x0C</td>
<td>Identification number byte 1</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x0D</td>
<td>Identification number byte 2</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x0E</td>
<td>Identification number byte 3</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x0F</td>
<td>Identification number byte 4</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x10</td>
<td>Identification number byte 5</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x11</td>
<td>Identification number byte 6</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x12</td>
<td>Identification number byte 7 (LSB)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x13</td>
<td>Hardware Version (major)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x14</td>
<td>Hardware Version (minor)</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x15</td>
<td>101 Version number of the current DALI standard</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x16</td>
<td>102 Version numbers of all integrated DALI control gears</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x17</td>
<td>103 Version numbers of all integrated DALI control devices</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x18</td>
<td>Number of logical control units in the device</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x19</td>
<td>Number of logical control gears in the device</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x1A</td>
<td>Index number of this logical DALI control gear or DALI control device</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0x1B – 0x7F</td>
<td>Reserved, not implemented</td>
<td></td>
</tr>
<tr>
<td>0x80 – 0xFE</td>
<td>Additional device information</td>
<td>Vendor-specific</td>
</tr>
<tr>
<td>0xFF</td>
<td>Reserved, not implemented</td>
<td></td>
</tr>
</tbody>
</table>

**Structure of memory bank 1:**

Memory bank 1 can be used by the device vendor to store further information in the DALI device. If this memory bank is used, all fields up to address 0x10 are occupied as follows.
<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Default values</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Address of the last memory area inside the memory bank that can be accessed.</td>
<td>Vendor-specific (0x10-0xFE)</td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>Indicator byte</td>
<td>Vendor-specific</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>Lock byte for memory bank 1. Writeable bytes become changeable through the value 0x55. No other values make writing possible.</td>
<td>0xFF</td>
<td></td>
</tr>
<tr>
<td>0x03</td>
<td>OEM GTIN byte0 (MSB)</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x04</td>
<td>OEM GTIN byte1</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x05</td>
<td>OEM GTIN byte2</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x06</td>
<td>OEM GTIN byte3</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x07</td>
<td>OEM GTIN byte4</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x08</td>
<td>OEM GTIN byte5 (LSB)</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x09</td>
<td>OEM Identification number byte 0 (MSB)</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x0A</td>
<td>OEM Identification number byte 1</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x0B</td>
<td>OEM Identification number byte 2</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x0C</td>
<td>OEM Identification number byte 3</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x0D</td>
<td>OEM Identification number byte 4</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x0E</td>
<td>OEM Identification number byte 5</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x0F</td>
<td>OEM Identification number byte 6</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x10</td>
<td>OEM Identification number byte 7 (LSB)</td>
<td>0xFF</td>
<td>Lockable by byte 0x02</td>
</tr>
<tr>
<td>0x11 – 0xFE</td>
<td>Additional device information</td>
<td>Vendor-specific</td>
<td></td>
</tr>
<tr>
<td>0xFF</td>
<td>Reserved, not implemented</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Structure of memory banks 2 to 199:**

The device vendor can use memory banks 2 to 199 to supply further parameters. The structure of the memory banks is always as shown below. The vendor of the DALI device must be consulted regarding the contents and the possibility to write individual bytes.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Default values</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Address of the last memory area inside the memory bank that can be accessed.</td>
<td>Vendor-specific (0x03 - 0xFE)</td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>Indicator byte</td>
<td>Vendor-specific</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>Lock byte for the memory bank. Writeable bytes become changeable through the value 0x55. No other values make writing possible.</td>
<td>0xFF</td>
<td></td>
</tr>
<tr>
<td>0x03 – 0xFE</td>
<td>Additional device information</td>
<td>Vendor-specific</td>
<td>Vendor-specific; lockable by byte 0x02 if enabled by the vendor</td>
</tr>
<tr>
<td>0xFF</td>
<td>Reserved, not implemented</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3 Programming

3.1 POUs

3.1.1 Applications

3.1.1.1 Part 102 (control gears)

3.1.1.1.1 Addressing

FB_DALI102Addressing

This function block addresses the DALI control gears at random. The user has no influence on which DALI control gear is assigned which short address. Short addresses are allocated in ascending order.

Applying a positive edge to the `bStart` input starts the function block, and the `bBusy` output goes TRUE. Depending on the selected options (parameter `nOptions`) the group membership and scenes are subsequently deleted. The terminal now addresses all DALI control gears independently. Once all DALI control gears have been addressed, the `bBusy` output goes back to FALSE. The `nAddressedDevices` output variable supplies information about how many DALI control gears have received a short address. Processing this function block can take several minutes, depending on how many DALI control gears are attached.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bStart</code></td>
<td>BOOL</td>
</tr>
<tr>
<td><code>nStartWithShortAddress</code></td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td><code>nOptions</code></td>
<td>DWORD</td>
</tr>
</tbody>
</table>

`bStart`: Execution of the DALI commands is triggered via a positive edge at this input.

`nStartWithShortAddress`: Short address, assigned to the first DALI control gear (0...63).

`nOptions`: Options for addressing the DALI control gears (see table below). The individual constants must be linked with OR operators.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI.GVL.cCompleteNewInstallation</td>
<td>All control gears are re-addressed, including control gears that already have a short address.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cDeleteAllGroupAssignments</td>
<td>Before addressing commences, any group assignments are deleted for all control gears, including those that are not actually addressed.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cDeleteAllSceneAssignments</td>
<td>Before addressing commences, any scene assignments are deleted for all control gears, including those that are not actually addressed.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOpticalFeedback</td>
<td>Prior to addressing, all control gears are set to minLevel. Newly addressed control gears are assigned maxLevel brightness after assignment of the short address.</td>
</tr>
</tbody>
</table>
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nAddressedDevices : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nAddressedDevices: If addressing has been completed (bBusy is FALSE), then the number of addressed control gears is shown at this output.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102ChangeAddresses

This function block can be used to change the short addresses of several DALI control gears.

A list of the DALI control gears for which the short address is to be changed is transferred in the array aChangeAddressList of type ST_DALIChangeAddressList [546]. The list has 64 entries from 0 to 63. Each entry contains a variable nOldAddress and nNewAddress with which the address assignment is parameterized. The end of the list is programmed with a 255 entry at nOldAddress, so that the whole list does not necessarily have to be filled in. If this entry is missing, however, then all entries are accepted. When the function block is started (positive edge on bStart), the list end is first determined on the basis of the described entry and afterwards the valid list range is examined for the following false entries:

- Address entries > 63
- Double address entry on the source page nOldAddress
- Double address entry on the target page nNewAddress (leads to double assignment of an address and, hence, to errors)

The function block then uses the short addresses to determine the internal long addresses of the DALI devices and enters them in the nRandomAddress parameter of the list.

If an error occurs during queries or during reprogramming, this leads to an error entry for the respective device in the list element nErrors (see ST_DALIChangeAddressList [546]).

The individual bits in the list element nErrors have the following meaning:
### Bit Error Table

<table>
<thead>
<tr>
<th>Bit</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error during reading of the long address</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Error during programming of the new short address</td>
</tr>
<tr>
<td>6</td>
<td><code>nOldAddress</code> and <code>nNewAddress</code> have the same values</td>
</tr>
</tbody>
</table>

#### VAR_INPUT

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nOptions**: Reserved for future expansions.

#### VAR_OUTPUT

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

#### VAR_IN_OUT

**aChangeAddressList**: List with the short addresses that are to be changed (see ST_DALIChangeAddressList [546]).

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
3.1.1.2 Power control

FB_DALI102ConstantLightControl

The FB_DALI102ConstantLightControl function block provides the basic functions for implementing constant light regulation.

The system attempts to regulate to a specified setpoint \((nSetpointValue)\) by dimming up and down cyclically. The control dynamics are determined by a dead time \((tDeadTime)\). The dead time defines the waiting time between the individual DALI commands for changing the output value \((nActualLevel)\). The smaller the dead time, the faster the adjustment. A freely definable hysteresis \((nHysteresis)\) prevents continuous oscillation around the setpoint. If the actual value is within the hysteresis range around the setpoint, the output value of the DALI control gears is not changed.

Operation

The function block offers the option to address a single DALI control gear via a short address, several DALI control gears via a group address, or all DALI control gears of a DALI line via a broadcast.

The variables \(minLevel \rightarrow 98\), \(maxLevel \rightarrow 98\), \(fadeTime \rightarrow 99\), \(extendedFadeTimeBase \rightarrow 99\) and \(extendedFadeTimeMultiplier \rightarrow 100\) are parameters that are stored separately in each DALI control gear. These variables can be changed in the DALI control gears by writing the properties and with a positive edge at the input \(bInitialize\).

Control

The output value of the DALI control gears can be switched by positive edges at the inputs \(bOn\), \(bOff\) and \(bToggle\). If the DALI control gears are switched on and control is active \((bEnable = TRUE)\), control is performed to the specified setpoint. If the control is not active \((bEnable = FALSE)\), the output value of the DALI control gears remains unchanged.

The step-by-step adaption of the output value of the DALI control gears is carried out with the DALI commands STEP UP and STEP DOWN. The dead time \((tDeadTime)\) specifies the time after which the output value is adapted. The DALI command STEP UP / STEP DOWN is called only once per control cycle. If the control deviation \((nDeviation)\) is smaller than \((nHysteresis / 2)\), the output value of the DALI control gears remains unchanged. The control deviation is calculated from \(nSetpointValue - nActualBrightness\).

Operation via the inputs \(bOn\), \(bOff\), and \(bToggle\) inputs

The output value of the DALI control gears can be changed immediately by positive edges at the inputs \(bOn\), \(bOff\), and \(bToggle\). This is independent of whether the control has been activated or deactivated \((bEnable)\).
The variables `fadeTime`, `extendedFadeTimeBase`, and `extendedFadeTimeMultiplier` specify the speed at which the output value is changed when the DALI control gears are switched on via `bOn` or `bToggle`. Switching off the DALI control gears by `bOff` or `bToggle` takes place immediately. In this case, `nSwitchOnLevel` is used as the switch-on value. This value must lie in the range of the properties `nMinLevel` and `nMaxLevel`.

**DALI short address reference device (nReferenceDeviceAddress)**

If several DALI control gears are addressed, the current output value of the reference DALI control gear is read out via `nReferenceDeviceAddress`. The DALI control gears are set to the desired output value, depending on the state of the reference DALI control gear.

The parameter `nReferenceDeviceAddress` is also used if the output value of the reference DALI control gear is read out cyclically in the background (`tCycleActualLevel > 0 sec`).

No reference DALI control gear is required if a single DALI control gear is to be controlled with the function block (`eAddressType = E_DALIAddressType.Short`). The output value of the individual DALI control gear is determined via `nAddress`. The parameter `nReferenceDeviceAddress` has no meaning in this case.

If `nReferenceDeviceAddress` is used, it must always contain the short address of a DALI control gear, which is also contained in the addressed DALI group. For a broadcast, a DALI control gear with the corresponding short address must be present on the DALI line.

**Example**

The following example shows how a DALI light sensor can be combined with the FB_DALIConstantLightControl function block. In this example, the PD11-BMS-FLAT DALI sensor from B.E.G. is used.

Since only the measured brightness is required, the instance for the motion sensor is deactivated (property `bEnableOccupancy`).

The brightness is read out from the DALI sensor via the input `bQueryBrightness`. Thus, the instance for the light sensor can also be deactivated (property `bEnableBrightness`).

At the end of the dead time, the output `bControlCyclerEnding` of the constant light regulation is set to TRUE. This positive edge is connected to the input `bQueryBrightness` of the DALI sensor. This means that the current brightness value is read out immediately before the control deviation (`nDeviation`) is calculated.

By deactivating both instances, the DALI sensor does not send any events and the DALI bus is not unnecessarily loaded.

The properties are initialized directly when the instance is declared. This means that explicit assignment at runtime is no longer necessary.

```plaintext
PROGRAM P_ConstantLightControl
VAR
  fbBrightness : FB_DALI_BEG_PD11_BMS_V8_Flat(Communication.fbKL6821Communication) :=
    (bEnableOccupancy := FALSE,
     bEnableBrightness := FALSE);
  fbConstantLightControl : FB_DALI102ConstantLightControl(Communication.fbKL6821Communication) :=
    (nMinLevel := 85,
     nMaxLevel := 254);
  bInitialize : BOOL;
  bToggle : BOOL;
  nBrightnessLevel : UINT;
  nActualLevel : BYTE;
  nDeviation : DINT;
END_VAR
```
VAR_INPUT

bInitialize : BOOL := FALSE;
bEnable : BOOL := TRUE;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nSetpointValue : UINT := 500;
nActualBrightness : UINT := 500;
nHysteresis : UINT := 30;
tDeadtime : TIME := T#10S;
bOn : BOOL;
bOff : BOOL;
bToggle : BOOL;
nSwitchOnLevel : BYTE := 254;
nReferenceDeviceAddress : BYTE := 0;
tCycleActualLevel : TIME := T#30S;
nOptions : DWORD := 0;

blinitialize: A positive edge at this input writes the values of all properties to the DALI control gears.

bEnable: Activates the constant light regulation as soon as this input is TRUE. If the input is FALSE, the constant light regulation is deactivated.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). The input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nSetpointValue: The setpoint is applied to this input (0…65535).

nActualBrightness: The actual value is applied to this input.

nHysteresis: Control hysteresis (1…65535) around the setpoint. If the actual value is within this range, the output values of the DALI control gears are not changed.
tDeadtime: After the dead time (T#3S...T#3600S) has elapsed, the control deviation is recalculated and, if necessary, the output values of the DALI control gears are reduced or increased by one stage.

bOn: A positive edge at this input sets the DALI control gears to nSwitchOnLevel.

bOff: The DALI control gears are switched off via a positive edge at this input.

bToggle: A positive edge at this input toggles the DALI control gears between Off and nSwitchOnLevel.

nSwitchOnLevel: Output value (minLevel...maxLevel) for switching on the DALI control gears via the inputs bOn and bToggle.

nReferenceDeviceAddress: Short address (0...63) of the reference DALI control gear for group call and broadcast. This parameter is not evaluated if eAddressType = E_DALIAddressType.Short. In this case, the reference DALI control gear is read out via nAddress.

tCycleActualLevel: Cycle time with which the current output value of the reference DALI control gear is read out in the background. Set the cycle time such that as few DALI commands as possible are sent. If the time is set to 0 sec, no readout takes place.

nOptions: Reserved for future extensions.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bInitializing : BOOL;
nActualLevel : BYTE;
nDeviation : DINT;
bControlCycleEnding : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bInitializing: The output is set as soon as the initialization of the DALI control gears has been started, and remains active until all DALI commands have been executed.

nActualLevel: Current output value (0, minLevel...maxLevel, 255) of the reference DALI control gear that is addressed via nReferenceDeviceAddress. If eAddressType = E_DALIAddressType.Short, the reference DALI control gear is read out via nAddress. A value of 255 (MASK) indicates that an error occurred when reading the output value from the reference DALI control gear. The cause may be, for example, a technical defect or the DALI control gear is in the start-up phase. In this case bError is not set to TRUE.

nDeviation: Actual control deviation (nSetpointValue - nActualBrightness).

bControlCycleEnding: Before the dead time (tDeadtime) expires, this output is set to TRUE for 2 seconds. This output can be used, for example, to read the current brightness from a DALI light sensor.

Properties

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Name | Type | Access | Initial value | Description
---|---|---|---|---
nMaxLevel | BYTE | Get, Set | 254 | See variable maxLevel [98].
nMinLevel | BYTE | Get, Set | 126 | See variable minLevel [98].
eFadeTime | E_DALIFadeTime | Get, Set | Disabled | See variable fadeTime [99].
eExtendedFadeTimeBase | E_DALIFadeTimeBase | Get, Set | Base01 | See variable extendedFadeTimeBase [99].
eExtendedFadeTimeMultiplier | E_DALIFadeTimeMultiplier | Get, Set | Disabled | See variable extendedFadeTimeMultiplier [100].
ipDALICommunication | I_DALICommunication | Get, Set | 0 | Interface pointer to the communication block (e.g. FB_KL6821Communication [343]).

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>

### FB_DALI102Dimmer1Switch

The function block FB_DALI102Dimmer1Switch provides the main functions for implementing a touch dimmer.

#### Operation

The input `bSwitch` is intended for linking with a push button. Pressing the button briefly switches the DALI control gears on or off. Pressing the button longer causes the output value of the DALI control gears to dim cyclically between the minimum and maximum values (see below).

The output value of the DALI control gears can be changed by positive edges at the inputs `bRecallMinLevel`, `bRecallMaxLevel`, `bOn`, `bOff`, `bToggle`, `bGoToScene` and `bSetLevel` (see below).
The function block offers the option to address a single DALI control gear via a short address, several DALI control gears via a group address or all DALI control gears of a DALI line via a broadcast.

The variables `minLevel`, `maxLevel`, `fadeRate`, `fadeTime`, `extendedFadeTimeBase` and `extendedFadeTimeMultiplier` are parameters that are stored separately in each DALI control gear. These variables can be changed in the DALI control gears via the properties and via the input `bInitialize`.

If DALI commands are sent for setting the output values (`bBusy = TRUE`), further positive edges at the inputs are ignored. New commands can be executed when `bBusy` has been reset.

**Operation via the input bSwitch**

A short signal at the input `bSwitch` switches the DALI control gears on or off, depending on the state of the reference DALI control gear. If the signal is present for longer, the system switches to dimmer mode and the output values of the DALI control gears move cyclically between `minLevel` and `maxLevel` until `bSwitch` is FALSE again. If the output value of the reference DALI control gear reaches the limit value `minLevel` or `maxLevel`, the function block changes its dimming direction.

If `bSwitch` is set to FALSE again, the current output values of the DALI control gears remain unchanged. A new pulse at the input `bSwitch` sets the output values to 0. If the input `bSwitch` is briefly deactivated in dimmer mode, the function block changes its dimming direction.

The rate at which the output values of the DALI control gears change in dimmer mode is determined by the variable `fadeRate` in the individual DALI control gears. The speed at which the DALI control gears are switched on again is specified by the variables `fadeTime`, `extendedFadeTimeBase` and `extendedFadeTimeMultiplier`.

**Operation via the inputs bOn, bOff, bGoToScene and bSetLevel**

The output value of the DALI control gears can be changed immediately by positive edges at the inputs `bOn`, `bOff`, `bGoToScene` and `bSetLevel`.

For `bOn`, `bGoToScene` and `bSetLevel` the speed at which the specified value is to be reached is specified by the variables `fadeTime`, `extendedFadeTimeBase` and `extendedFadeTimeMultiplier`. `bOff` switches the DALI control gears off immediately.

**Memory mode**

When switching on, a distinction must be made as to whether memory mode (see `nOptions`) is active or not. If memory mode is active, the last set value is adopted as the output value for the DALI control gears when the device is switched on. If memory mode is not active, the output value from parameter `nLevelMemoryMode` is used for the DALI control gears. It is irrelevant whether the DALI control gears are switched via the input `bOn`, `bToggle` or `bSwitch`.

**DALI short address reference device (nReferenceDeviceAddress)**

If several DALI control gears are addressed, the current output value of the reference DALI control gear is read out via `nReferenceDeviceAddress`. The DALI control gears are set to the desired value, depending on the state of the reference DALI control gear.

The parameter `nReferenceDeviceAddress` is also used if the output value of the reference DALI control gear is read out cyclically in the background (`tCycleActualLevel > 0 sec`).

No reference DALI control gear is required if a single DALI control gear is to be controlled with the function block (`eAddressType = E_DALIAddressType.Short`). The output value of the individual DALI control gear is determined via `nAddress`. The parameter `nReferenceDeviceAddress` has no meaning in this case.

If `nReferenceDeviceAddress` is used, it must always contain the short address of a DALI control gear, which is also contained in the addressed DALI group. For a broadcast a DALI control gear with the corresponding short address must be present on the DALI line.

**VAR_INPUT**
bInitialize := BOOL := FALSE;
nAddress := BYTE;
eAddressType := E_DALIAddressType := E_DALIAddressType.Short;
bSwitch := BOOL;
bRecallMaxLevel := BOOL;
bRecallMinLevel := BOOL;
bOn := BOOL;
bOff := BOOL;
bToggle := BOOL;
bGoToScene := BOOL;
nScene := BYTE;
bSetLevel := BOOL;
nLevel := BYTE := 254;
nLevelMemoryMode := BYTE := 254;
nReferenceDeviceAddress := BYTE;
tCycleActualLevel := TIME := T#30S;
nOptions := DWORD := 0;

bInitialize: A positive edge at this input writes the values of all properties to the DALI control gears.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). The input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

bSwitch: A short signal at this input switches the DALI control gears on or off. If the signal is present for a longer period, the system switches to dimmer mode and the output values of the DALI control gears move cyclically between minLevel and maxLevel.

bRecallMaxLevel: A positive edge at this input sets the DALI control gears to maxLevel.

bRecallMinLevel: A positive edge at this input sets the DALI control gears to minLevel.

bOn: If memory mode is active (see nOptions), a positive edge at this input sets the DALI control gears to their output value before the last switching off. If memory mode is not active, the DALI control gears are set to nLevelMemoryMode.

bOff: The DALI control gears are switched off via a positive edge at this input. The previous output value is stored internally to be used for switching on if memory mode is active (see nOptions).

bToggle: Each positive edge at this input causes the DALI control gears to switch between off and their output value before the last switching off (memory mode active) or nLevelMemoryMode (memory mode not active).

bGoToScene: A positive edge at this input sets the output value of the DALI control gears to the value stored in the nScene scene.

nScene: Scene (0...15) to be called up in the DALI control gears by bGoToScene. Each DALI control gear has its own output value for each scene.

bSetLevel: A positive edge at this input sets the output value of the DALI control gears to the value specified by nLevel.

nLevel: Output value (0, minLevel...maxLevel, 255) to be called up in the DALI control gears by bSetLevel. The output value remains unchanged if the value is 255 (MASK).

nLevelMemoryMode: Output value (minLevel...maxLevel) for switching on the DALI control gears when memory mode is not active.

nReferenceDeviceAddress: Short address (0…63) of the reference DALI control gear for group call and broadcast. This parameter is not evaluated if eAddressType = E_DALIAddressType.Short. In this case, the reference DALI control gear is read out via nAddress.

tCycleActualLevel: Cycle time with which the current output value of the reference DALI control gear is read out in the background. Set the cycle time such that as few DALI commands as possible are sent. If the time is set to 0 sec, no readout takes place.

nOptions: Options that affect the behavior of the function block. The individual constants must be linked with OR operators.
### Constant

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI.GVL.cMemoryMode</td>
<td>Activates memory mode</td>
</tr>
</tbody>
</table>

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

- **bInitializing**: The output is set as soon as the initialization of the DALI control gears has been started, and remains active until all DALI commands have been executed.

- **nActualLevel**: Current output value (0, $minLevel...maxLevel$, 255) of the reference DALI control gear that is addressed via `nReferenceDeviceAddress`. If `eAddressType` is E_DALIAddressType.Short the reference DALI control gear is read out via `nAddress`. A value of 255 (MASK) indicates that an error occurred while reading the output value from the reference DALI control gear. The cause could be, for example, a technical defect or that the DALI control gear is in the start-up phase. In this case `bError` is not set to TRUE.

### PROPERTIES

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

#### Name | Type | Access | Initial value | Description
---      |------|--------|---------------|-------------------|
| nMaxLevel | BYTE | Get, Set | 254 | See variable maxLevel [98]. |
| nMinLevel  | BYTE | Get, Set | 126 | See variable minLevel [98]. |
| eFadeRate  | E_DALIFadeRate | Get, Set | N045StepsPerSec | See variable fadeRate [98]. |
| eFadeTime  | E_DALIFadeTime | Get, Set | Disabled | See variable fadeTime [99]. |
| eExtendedFadeTimeBase | E_DALIExtendedFadeTimeBase | Get, Set | Base01 | See variable extendedFadeTimeBase [99]. |
| eExtendedFadeTimeMultiplier | E_DALIExtendedFadeTimeMultiplier | Get, Set | Disabled | See variable extendedFadeTimeMultiplier [100]. |
| ipDALICommunication | I.DALICOmmunication | Get, Set | 0 | Interface pointer to the communication block (e.g. FB_KL6821Communication [343]). |

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
FB_DALI102Dimmer2Switch

The function block FB_DALI102Dimmer2Switch provides the main functions for implementing a touch dimmer.

**Operation**

The inputs $bSwitchUp$ and $bSwitchDown$ are intended for linking with two push buttons. Pressing the button briefly switches the DALI control gears on or off. Pressing the button longer causes the output value of the DALI control gears to dim down to the minimum or up to the maximum values, respectively (see below).

The output value of the DALI control gears can be changed by positive edges at the inputs $bRecallMinLevel$, $bRecallMaxLevel$, $bOn$, $bOff$, $bToggle$, $bGoToScene$, and $bSetLevel$ (see below).

The function block offers the option to address a single DALI control gear via a short address, several DALI control gears via a group address or all DALI control gears of a DALI line via a broadcast.

The variables $minLevel$, $maxLevel$, $fadeRate$, $fadeTime$, $extendedFadeTimeBase$ and $extendedFadeTimeMultiplier$ are parameters that are stored separately in each DALI control gear. These variables can be changed in the DALI control gears by writing the properties and with a positive edge at the input $bInitialize$.

If DALI commands are sent for setting the output values ($bBusy$ = TRUE), further positive edges at the inputs are ignored. New commands can be executed as soon as $bBusy$ has been reset again.

**Operation via the $bSwitchUp$ and $bSwitchDown$ inputs**

A short signal at the inputs $bSwitchUp$ or $bSwitchDown$ switches the DALI control gears on or off, depending on the state of the reference DALI control gear. If the signal is present for a longer time, the system switches to dimmer mode and the output values of the DALI control gears move up to $minLevel$ or $maxLevel$ or until $bSwitchUp$ or $bSwitchDown$ are FALSE again.

If $bSwitchUp$ or $bSwitchDown$ is set to FALSE again, the current output values of the DALI control gears remain unchanged. A new pulse at the input $bSwitchUp$ or $bSwitchDown$ sets the output values to 0.

The rate at which the output values of the DALI control gears change in dimmer mode is determined by the variable $fadeRate$ in the individual DALI control gears. The speed at which the DALI control gears are switched on again is specified by the variables $fadeTime$, $extendedFadeTimeBase$, and $extendedFadeTimeMultiplier$.

**Operation via the inputs $bOn$, $bOff$, $bGoToScene$, and $bSetLevel$**

The output value of the DALI control gears can be changed immediately by positive edges at the inputs $bOn$, $bOff$, $bGoToScene$, and $bSetLevel$. 
For \( b\text{On} \), \( b\text{GoToScene} \), and \( b\text{SetLevel} \), the speed at which the specified value is to be reached is specified by the variables \( \text{fadeTime} \), \( \text{extendedFadeTimeBase} \), and \( \text{extendedFadeTimeMultiplier} \). \( b\text{Off} \) switches the DALI control gears off immediately.

### Memory mode

When switching on, a distinction must be made as to whether memory mode (see \( n\text{Options} \)) is active or not. If memory mode is active, the last set value is adopted as the output value for the DALI control gears when the device is switched on. If memory mode is not active, the output value from the parameter \( n\text{LevelMemoryMode} \) is used for the DALI control gears. It is irrelevant whether the DALI control gears are switched via the input \( b\text{On} \), \( b\text{Toggle} \), \( b\text{SwitchUp} \), or \( b\text{SwitchDown} \).

### DALI short address reference device (\( n\text{ReferenceDeviceAddress} \))

If several DALI control gears are addressed, the current output value of the reference DALI control gear is read out via \( n\text{ReferenceDeviceAddress} \). The DALI control gears are set to the desired value, depending on the state of the reference DALI control gear.

The parameter \( n\text{ReferenceDeviceAddress} \) is also used if the output value of the reference DALI control gear is read out cyclically in the background (\( t\text{CycleActualLevel} > 0 \) sec).

No reference DALI control gear is required if a single DALI control gear is to be controlled with the function block \( \text{E_DALIAddressType} = \text{E_DALIAddressType.Short} \). The output value of the individual DALI control gear is determined via \( n\text{Address} \). The parameter \( n\text{ReferenceDeviceAddress} \) has no meaning in this case.

If \( n\text{ReferenceDeviceAddress} \) is used, it must always contain the short address of a DALI control gear, which is also contained in the addressed DALI group. For a broadcast, a DALI control gear with the corresponding short address must be present on the DALI line.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b\text{Initialize} )</td>
<td>A positive edge at this input writes the values of all properties to the DALI control gears.</td>
</tr>
<tr>
<td>( n\text{Address} )</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>( e\text{AddressType} )</td>
<td>Defines whether the input ( n\text{Address} ) contains a short address (0…63) or a group address (0…15). The input ( n\text{Address} ) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \text{E_DALIAddressType}</td>
</tr>
<tr>
<td>( b\text{SwitchUp} )</td>
<td>A short signal at these inputs switches the DALI control gears on or off. If the signal is present for a longer period, the system switches to dimmer mode and the output values of the DALI control gears move to \text{minLevel} or \text{maxLevel}.</td>
</tr>
<tr>
<td>( b\text{RecallMaxLevel} )</td>
<td>A positive edge at this input sets the DALI control gears to \text{maxLevel}.</td>
</tr>
<tr>
<td>( b\text{RecallMinLevel} )</td>
<td>A positive edge at this input sets the DALI control gears to \text{minLevel}.</td>
</tr>
<tr>
<td>( b\text{On} )</td>
<td>If memory mode is active (see ( n\text{Options} )), a positive edge at this input sets the DALI control gears to their output value before the last switching off. If memory mode is not active, the DALI control gears are set to ( n\text{LevelMemoryMode} ).</td>
</tr>
</tbody>
</table>
bOff: The DALI control gears are switched off via a positive edge at this input. The previous output value is stored internally to be used for switching on if memory mode is active (see nOptions).

bToggle: Each positive edge at this input causes the DALI control gears to switch between off and their output value before the last switching off (memory mode active) or nLevelMemoryMode (memory mode not active).

bGoToScene: A positive edge at this input sets the output value of the DALI control gears to the value stored in the nScene scene.

defined SCENE (0...15) to be called up in the DALI control gears by bGoToScene. Each DALI control gear has its own output value for each scene.

bSetLevel: A positive edge at this input sets the output value of the DALI control gears to the value specified by nLevel.

nLevel: Output value (0, minLevel...maxLevel, 255) to be called up in the DALI control gears by bSetLevel. The output value remains unchanged if the value is 255 (MASK).

nLevelMemoryMode: Output value (minLevel...maxLevel) for switching on the DALI control gears when memory mode is not active.

nReferenceDeviceAddress: Short address (0...63) of the reference DALI control gear for group call and broadcast. This parameter is not evaluated if eAddressType = E_DALIAddressType.Short. In this case, the reference DALI control gear is read out via nAddress.

tCycleActualLevel: Cycle time with which the current output value of the reference DALI control gear is read out in the background. Set the cycle time such that as few DALI commands as possible are sent. If the time is set to 0 sec, no readout takes place.

nOptions: Options that affect the behavior of the function block. The individual constants must be linked with OR operators.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI.GVL.cMemoryMode</td>
<td>Activates memory mode.</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bInitializing : BOOL;
nActualLevel : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bInitializing: The output is set as soon as the initialization of the DALI control gears has been started, and remains active until all DALI commands have been executed.

nActualLevel: Current output value (0, minLevel...maxLevel, 255) of the reference DALI control gear that is addressed via nReferenceDeviceAddress. If eAddressType = E_DALIAddressType.Short, the reference DALI control gear is read out via nAddress. A value of 255 (MASK) indicates that an error occurred when reading the output value from the reference DALI control gear. The cause may be, for example, a technical defect or the DALI control gear is in the start-up phase. In this case bError is not set to TRUE.

Properties

All parameters that are written to the DALI control device via bInitialize are available as properties.
**Programming**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nMaxLevel</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>254</td>
<td>See variable <code>maxLevel</code> [98].</td>
</tr>
<tr>
<td>nMinLevel</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>126</td>
<td>See variable <code>minLevel</code> [98].</td>
</tr>
<tr>
<td>eFadeRate</td>
<td>E_DALIFadeRate [553]</td>
<td>Get, Set</td>
<td>N045StepsPerSec</td>
<td>See variable <code>fadeRate</code> [98].</td>
</tr>
<tr>
<td>eFadeTime</td>
<td>E_DALIFadeTime [553]</td>
<td>Get, Set</td>
<td>Disabled</td>
<td>See variable <code>fadeTime</code> [99].</td>
</tr>
<tr>
<td>eExtendedFadeTimeBase</td>
<td>E_DALIExtendedFadeTimeBase [552]</td>
<td>Get, Set</td>
<td>Base01</td>
<td>See variable <code>extendedFadeTimeBase</code> [99].</td>
</tr>
<tr>
<td>eExtendedFadeTimeMultiplier</td>
<td>E_DALIExtendedFadeTimeMultiplier [552]</td>
<td>Get, Set</td>
<td>Disabled</td>
<td>See variable <code>extendedFadeTimeMultiplier</code> [100].</td>
</tr>
<tr>
<td>ipDALICommunication</td>
<td>I_DALICommunication</td>
<td>Get, Set</td>
<td>0</td>
<td>Interface pointer to the communication block (e.g. <code>FB_KL6821Communication</code> [343]).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>

**3.1.1.1.3 Settings**

**FB_DALI102GetSettings**

This function block reads several variables (see table below [27]) from selected DALI control gears and saves them in the structure `ST_DALIControlGearSettings` [546].

The DALI control gears that are to be read can be specified by the input variable `nDevices`. Each bit of this variable corresponds to a short address.

Applying a positive edge to the `bStart` input starts the function block, and the `bBusy` output goes TRUE. A check is first made as to whether a DALI control gear is present at all. If this is the case, then the `bPresent` bit is set in the respective structure (see `ST_DALIControlGearSettings` [546]), after which the settings of the control gear are read one by one and written in the associated variables in the structure. If it is found that a device is not available, the reading is skipped and work continues with the next device. The structure index here reflects the address of the device. In other words, data for the device with short address 0 is located at `arrDALIDeviceSettings[0]`, and so on through to the device with short address 63 having its data at...
arrDALIDeviceSettings[63]. If a read error occurs when reading from a device, the corresponding bit in nErrors is set for the respective structure without the function block itself switching to error mode. The following table shows which bit is set in the nErrors variable when an error occurs during the reading of a variable from a control gear.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>An error occurred while attempting to seek the DALI control gear.</td>
</tr>
<tr>
<td>1</td>
<td>Error while reading the variable actualLevel[98].</td>
</tr>
<tr>
<td>2</td>
<td>Error while reading the variable powerOnLevel[98].</td>
</tr>
<tr>
<td>3</td>
<td>Error while reading the variable systemFailureLevel[98].</td>
</tr>
<tr>
<td>4</td>
<td>Error while reading the variable minLevel[98].</td>
</tr>
<tr>
<td>5</td>
<td>Error while reading the variable maxLevel[98].</td>
</tr>
<tr>
<td>6</td>
<td>Error while reading the variable fadeRate[98].</td>
</tr>
<tr>
<td>7</td>
<td>Error while reading the variable fadeTime[99].</td>
</tr>
<tr>
<td>8</td>
<td>Error while reading the variables extendedFadeTimeBase[99] and extendedFadeTimeMultiplier[100].</td>
</tr>
<tr>
<td>9</td>
<td>Error while reading the variable randomAddress[101].</td>
</tr>
<tr>
<td>10</td>
<td>Error while reading the variable gearGroups[102].</td>
</tr>
<tr>
<td>11</td>
<td>Error while reading the variable scene0-scene15[102].</td>
</tr>
<tr>
<td>12</td>
<td>Error while reading the variable statusInformation[83].</td>
</tr>
<tr>
<td>13</td>
<td>Error while reading the version number from the MemoryBank[9].</td>
</tr>
<tr>
<td>14</td>
<td>Error while reading the supported device types.</td>
</tr>
<tr>
<td>15</td>
<td>Error while reading the variable physicalMinLevel[102].</td>
</tr>
</tbody>
</table>

**VAR_INPUT**

bStart : BOOL;
nDevices : LWORD;
nOptions : DWORD;
bCancel : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nDevices:** Variable for selecting the DALI control gears to be read. Each bit of the variable represents a DALI control gear (0...63). All short addresses are queried with a setting of 16#FFFF_FFFF_FFFF_FFFF.

**nOptions:** Options for reading the variables (see table below). The individual constants must be linked with OR operators.
<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI.GVL.cOptionActualDimLevel</td>
<td>Reads the current output value [98].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionPowerOnLevel</td>
<td>Reads the current switch-on value [98].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionSystemFailureLevel</td>
<td>Reads the current setting value [98] in case of a DALI bus error.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionMinLevel</td>
<td>Reads the maximum [98] output value.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionMaxLevel</td>
<td>Reads the minimum [98] output value.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionFadeRateFadeTime</td>
<td>Reads the FadeRate [98] and the FadeTime [99].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionExtendedFadeTime</td>
<td>Reads the extended FadeTimeBase [99] and the FadeTimeMultiplier [100].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionRandomAddress</td>
<td>Reads the random address [101].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionGroups</td>
<td>Reads the group allocations [102].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionSceneLevels</td>
<td>Reads the scene settings [102].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionStatusInformation</td>
<td>Reads the status information [83].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionVersionNumber</td>
<td>Reads the version number [9] of the DALI control gear.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionDeviceType</td>
<td>Reading the device type [73].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionPhysicalMinLevel</td>
<td>Reads the PhysicalMinLevel [102].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionAll</td>
<td>Reads all variables.</td>
</tr>
</tbody>
</table>

bCancel: A positive edge at this input will disable the function block and hence abort the reading of the variables.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are processed (see E_DALICommandPriority [549]).

**eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>ITcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nCurrentShortAddress</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nCurrentShortAddress**: Indicates which short address is currently being queried.

VAR_IN_OUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrDALIDeviceSettings</td>
<td>ARRAY [0..63] of ST_DALIControlGearSettings;</td>
</tr>
</tbody>
</table>

**arrDALIDeviceSettings**: Array containing the read parameters (see ST_DALIControlGearSettings [546]) of the individual control gears.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

FB_DALI102SetSettings

This function block writes the values stored in the structure `ST_DALIControlGearSettings` to several variables (see table below) of selected DALI control gears in a DALI line.

The DALI control gears that are to be written to can be specified by the input variable `nDevices`. Each bit of this variable corresponds to a short address.

Applying a positive edge to the `bStart` input starts the function block, and the `bBusy` output goes TRUE. A check is first made as to whether a DALI control gear is present at all. If this is the case, then the `bPresent` bit is set in the respective structure (see `ST_DALIControlGearSettings`), after which the settings are written one by one to the control gear. If a device is detected as being missing, the writing is skipped and work continues with the next device. The structure index here reflects the address of the device. In other words, data for the device with short address 0 is located at `arrDALIDeviceSettings[0]`, and so on through to the device with short address 63 having its data at `arrDALIDeviceSettings[63]`. If a device error occurs during writing, the corresponding bit in `nErrors` is set for the respective structure without the function block itself switching to error mode. The following table shows which bit is set in the `nErrors` variable when an error occurs during the reading of a variable from a control gear.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>An error occurred while attempting to seek the DALI control gear.</td>
</tr>
<tr>
<td>2</td>
<td>Error while writing the variable <code>powerOnLevel</code></td>
</tr>
<tr>
<td>3</td>
<td>Error while writing the variable <code>systemFailureLevel</code></td>
</tr>
<tr>
<td>4</td>
<td>Error while writing the variable <code>minLevel</code></td>
</tr>
<tr>
<td>5</td>
<td>Error while writing the variable <code>maxLevel</code></td>
</tr>
<tr>
<td>6</td>
<td>Error while writing the variable <code>fadeRate</code></td>
</tr>
<tr>
<td>7</td>
<td>Error while writing the variable <code>fadeTime</code></td>
</tr>
<tr>
<td>8</td>
<td>Error while writing the variables <code>extendedFadeTimeBase</code> and <code>extendedFadeTimeMultiplier</code></td>
</tr>
<tr>
<td>10</td>
<td>Error while writing the variable <code>gearGroups</code></td>
</tr>
<tr>
<td>11</td>
<td>Error while writing the variable <code>scene0-scene15</code></td>
</tr>
</tbody>
</table>

VAR_INPUT

```plaintext
VAR_INPUT
bStart : BOOL;
nDevices : LWORD;
nOptions : DWORD;
bCancel : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
```

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nDevices: Variable for selecting the DALI control gears that are to be initialized. Each bit of the variable represents a DALI control gear (0..63). All short addresses are written to with a setting of 16#FFFF_FFFF_FFFF_FFFF.
nOptions: Options for writing the variables (see table below). The individual constants must be linked with OR operators.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI.GVL.cOptionPowerOnLevel</td>
<td>Initializes the current switch-on value [98].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionSystemFailureLevel</td>
<td>Initializes the current setting value [98] in case of a DALI bus error.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionMinLevel</td>
<td>Initializes the maximum [98] output value.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionMaxLevel</td>
<td>Initializes the minimum [98] output value.</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionFadeRate</td>
<td>Initializes the FadeRate [98].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionFadeTime</td>
<td>Initializes the FadeTime [99].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionExtendedFadeTime</td>
<td>Initializes the extended FadeTimeBase [99] and the FadeTimeMultiplier [100].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionGroups</td>
<td>Initializes the group allocations [102].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionSceneLevels</td>
<td>Initializes the scene settings [102].</td>
</tr>
<tr>
<td>Tc3_DALI.GVL.cOptionAll</td>
<td>Initializes all variables.</td>
</tr>
</tbody>
</table>

bCancel: A positive edge at this input will disable the function block and hence abort the initialization of the variable.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

  
eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nCurrentShortAddress</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nCurrentShortAddress: Indicates which short address is currently being written to.

VAR_IN_OUT

arrDALIDeviceSettings : ARRAY [0..63] of ST_DALIControlGearSettings;

arrDALIDeviceSettings: Array containing the read parameters (see ST_DALIControlGearSettings [546]) of the individual control gears.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>
FB_DALI102ReadMemoryBank

The specified memory bank of the DALI control gear is read (see Memory banks [9]). The memory bank is specified by the parameter nMemoryBank. The range to be read can be limited with the help of the parameters nSubRangeStart and nSubRangeEnd.

The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

VAR_INPUT

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bCancel</td>
<td>BOOL</td>
<td>A rising edge at this input disables the function block and aborts the reading of the variables.</td>
</tr>
<tr>
<td>nMemoryBank</td>
<td>BYTE</td>
<td>Specifies the memory bank for the read access.</td>
</tr>
<tr>
<td>nSubRangeStart</td>
<td>BYTE (0..254)</td>
<td>Offset inside the memory bank from which reading is to take place.</td>
</tr>
<tr>
<td>nSubRangeEnd</td>
<td>BYTE</td>
<td>Offset inside the memory bank up to which reading is to take place.</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

nOptions: Reserved for future extensions.

bCancel: A rising edge at this input disables the function block and aborts the reading of the variables.

nMemoryBank: Specifies the memory bank for the read access.

nSubRangeStart: Offset inside the memory bank from which reading is to take place.

nSubRangeEnd: Offset inside the memory bank up to which reading is to take place.

nSubRangeEnd Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>Offset 0 of the memory bank determines the offset up to which reading is to take place.</td>
</tr>
<tr>
<td>254</td>
<td>Offset 0 is not read at the beginning. Reading always takes place up to Offset 254.</td>
</tr>
<tr>
<td>&lt;= 254</td>
<td>nSubRangeEnd may not be greater than the value in Offset 0. The partial areas of the memory bank are read with this setting.</td>
</tr>
</tbody>
</table>
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nCurrentOffset : BYTE;
nLastOffset : BYTE;
aValues : ARRAY [0..254] OF BYTE;
aErrors : ARRAY [0..254] OF BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nCurrentOffset: The offset that is currently being read is output at this output.

nLastOffset: Last valid offset of the memory bank that was read.

aValues: Array containing the read parameters of the memory bank.

aErrors: Array containing errors of the parameters of the memory bank that occurred during reading.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.1.2 Part 103 (control devices)

3.1.1.2.1 Addressing

FB_DALI103Addressing

This function block addresses the DALI control devices at random. The user has no influence on which control device is assigned which short address. Short addresses are allocated in ascending order.

Applying a positive edge to the bStart input starts the function block, and the bBusy output goes TRUE. The terminal now addresses all DALI control devices independently. Once all DALI control devices have been addressed, the bBusy output goes back to FALSE. The output variable nAddressedDevices provides information on how many DALI control devices were assigned a short address. Depending on how many DALI control devices are connected, processing of this function block can take several minutes.

The function block cannot be used when using the KL6811.
VAR_INPUT

bStart : BOOL;
nStartWithShortAddress : BYTE := 0;
nOptions : DWORD := Tc3.DALI.GVL.cOpticalFeedback;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nStartWithShortAddress: Short address assigned to the first DALI control device (0…63).

nOptions: Options for addressing the DALI control devices (see table below). The individual constants must be linked with OR operators.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3.DALI.GVL.cCompleteNewInstallation</td>
<td>All DALI control devices are readdressed, even those that already have a short address.</td>
</tr>
<tr>
<td>Tc3.DALI.GVL.cOpticalFeedback</td>
<td>Newly addressed DALI control devices are allocated the DALI command IDENTIFY DEVICE once the short address has been assigned.</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nAddressedDevices : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nAddressedDevices: When addressing is completed (bBusy is FALSE), the number of addressed control devices is displayed at this output.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3.DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103ChangeAddresses

This function block can be used to change the short addresses of several DALI control devices.

A list of the DALI control devices for which the short address is to be changed is transferred in the array aChangeAddressList of type ST_DALIChangeAddressList [546]. The list has 64 entries from 0 to 63. Each entry contains a variable nOldAddress and nNewAddress with which the address assignment is parameterized. The end of the list is programmed with a 255 entry at nOldAddress, so that the whole list does not necessarily have to be filled in. If this entry is missing, however, then all entries are accepted.

When the function block is started (positive edge on bStart), the list end is first determined on the basis of the described entry and afterwards the valid list range is examined for the following false entries:

- Address entries > 63
- Double address entry on the source page nOldAddress
• Double address entry on the target page \texttt{nNewAddress} (leads to double assignment of an address and, hence, to errors)

The function block then uses the short addresses to determine the internal long addresses of the DALI devices and enters them in the \texttt{nRandomAddress} parameter of the list.

If an error occurs during queries or during reprogramming, this leads to an error entry for the respective device in the list element \texttt{nErrors} (see \texttt{ST_DALIChangeAddressList} [546]).

The individual bits in the list element \texttt{nErrors} have the following meaning:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error during reading of the long address</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Error during programming of the new short address</td>
</tr>
<tr>
<td>6</td>
<td>\texttt{nOldAddress} and \texttt{nNewAddress} have the same values</td>
</tr>
</tbody>
</table>

The function block cannot be used when using the KL6811.

\begin{itemize}
  \item \textbf{VAR\_INPUT}
  \begin{itemize}
    \item \texttt{bStart} : \texttt{BOOL};
    \item \texttt{nOptions} : \texttt{DWORD := 0};
  \end{itemize}
  \begin{itemize}
    \item \texttt{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.
    \item \texttt{nOptions}: Reserved for future expansions.
  \end{itemize}

  \begin{itemize}
    \item \textbf{VAR\_OUTPUT}
    \begin{itemize}
      \item \texttt{bError} : \texttt{BOOL};
      \item \texttt{ipResultMessage} : \texttt{I_TcMessage};
      \item \texttt{bBusy} : \texttt{BOOL};
    \end{itemize}
    \begin{itemize}
      \item \texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.
      \item \texttt{ipResultMessage}: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.
      \item \texttt{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
    \end{itemize}

  \begin{itemize}
    \item \textbf{VAR\_IN\_OUT}
    \begin{itemize}
      \item \texttt{aChangeAddressList} : ARRAY [0..63] OF \texttt{ST_DALIChangeAddressList};
    \end{itemize}
    \begin{itemize}
      \item \texttt{aChangeAddressList}: List with the short addresses that are to be changed (see \texttt{ST_DALIChangeAddressList} [546]).
    \end{itemize}

  \begin{itemize}
    \item \textbf{Requirements}
    \begin{itemize}
      \item Development environment: TwinCAT from v3.1.4022.29
      \item required PLC library: Tc3\_DALI from v3.1.4.0
    \end{itemize}
\end{itemize}
3.1.1.2.2 Settings

FB_DALI103ReadMemoryBank

The specified memory bank of the DALI control device is read (see Memory banks [9]). The memory bank is specified by the parameter nMemoryBank. The range to be read can be limited with the help of the parameters nSubRangeStart and nSubRangeEnd.

The function block changes the DTR0 and the DTR1 of all connected DALI control devices.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
nOptions : DWORD := 0;
bCancel : BOOL;
nMemoryBank : BYTE := 0;
nSubRangeStart : BYTE (0..254) := 0;
nSubRangeEnd : BYTE := 255;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (high, middle or low) with which the DALI commands are processed (see E_DALICommandPriority [549]).

nOptions: Reserved for future extensions.

bCancel: A rising edge at this input disables the function block and aborts the reading of the variables.

nMemoryBank: Specifies the memory bank for the read access.

nSubRangeStart: Offset inside the memory bank from which reading is to take place.

nSubRangeEnd: Offset inside the memory bank up to which reading is to take place.
**nSubRangeEnd**  |  Description
--- | ---
255  | Offset 0 of the memory bank determines the offset up to which reading is to take place.
254  | Offset 0 is not read at the beginning. Reading always takes place up to Offset 254.
<=254  | **nSubRangeEnd** may not be greater than the value in Offset 0. The partial areas of the memory bank are read with this setting.

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError : BOOL;</td>
</tr>
<tr>
<td>ipResultMessage : I_TcMessage;</td>
</tr>
<tr>
<td>bBusy : BOOL;</td>
</tr>
<tr>
<td>nCurrentOffset : BYTE;</td>
</tr>
<tr>
<td>nLastOffset : BYTE;</td>
</tr>
<tr>
<td>aValues : ARRAY [0..254] OF BYTE;</td>
</tr>
<tr>
<td>aErrors : ARRAY [0..254] OF BOOL;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nCurrentOffset**: The offset that is currently being read is output at this output.

**nLastOffset**: Last valid offset of the memory bank that was read.

**aValues**: Array containing the read parameters of the memory bank.

**aErrors**: Array containing errors of the parameters of the memory bank that occurred during reading.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### 3.1.2 Commands

#### 3.1.2.1 Part 102 (control gears)

#### 3.1.2.1.1 Application extended commands

**FB_DALI102QueryExtendedVersionNumber**

The *extendedVersionNumber* variable is read from the DALI control gear. It specifies the version for the implemented, application-related extension commands.
This command belongs to the application-related extension commands.

Since each DALI control gear can support several device types, the DALI command ENABLE DEVICE TYPE must be called before each application-related extension command. Application-related extension commands are defined in the IEC 62386-2xx standard.

The function block FB_DALI102QueryExtendedVersionNumber independently executes the DALI command ENABLE DEVICE TYPE internally.

The input variable nDeviceType specifies for which device type the variable extendedVersionNumber is to be read.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI device or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]). eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.</td>
</tr>
<tr>
<td>eDeviceType</td>
<td>Identification of the device type (see E_DALIDeviceType [550]).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nExtendedVersionNumber</td>
<td>Extended version number of the control gear.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
3.1.2.1.2 Configuration commands

FB_DALI102AddToGroup

The function block adds the DALI control gear to the $nGroup$ group.

Internally, the group membership is stored in the variable $gearGroups[102]$.

Each bit of $gearGroups$ represents a group. Bit 0 corresponds to group 0 etc., up to bit 15 in group 15. If a bit is set, the DALI control gear belongs to the corresponding group.

The function block sends the DALI command ADD TO GROUP twice, as required by the DALI standard for certain commands.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nGroup : BYTE;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

   eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nGroup: Group number (0…15).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
busy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102IdentifyDevice**

The function block starts the identification routine for the DALI control gear.

It takes approx. 10 seconds and ends automatically. The exact scope of the identification routine depends on the manufacturer of the control gear.

---

**VAR_INPUT**

bStart : BOOL; 
nAddress : BYTE; 
eAddressType : E_DALIAddressType := E_DALIAddressType.Short; 
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).

- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

---

**VAR_OUTPUT**

bError : BOOL; 
ipResultMessage : I_TcMessage; 
bBusy : BOOL;

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB DALI102RemoveFromGroup**

The function block removes the DALI control gear from the group `nGroup`.

Internally, the group membership is stored in the variable `gearGroups[102]`.

Each bit of `gearGroups` represents a group. Bit 0 corresponds to group 0 etc., up to bit 15 in group 15. If a bit is set, the DALI control gear belongs to the corresponding group.

- **VAR_INPUT**
  - `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
  - `nAddress`: Address of a DALI device or a DALI group.
  - `eAddressType`: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType[547]`).
  - `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority[549]`). `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
  - `nGroup`: Group number (0…15).

- **VAR_OUTPUT**
  - `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
  - `ipResultMessage`: Interface pointer (see Error evaluation[574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages[561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102_RemoveFromScene

The function block removes the DALI control gear from the scene nScene.

Internally, the scenes are stored in the variables scene0 to scene15 (see scene0-scene15 [102]).

The function block sends the DALI command REMOVE FROM SCENE twice, as required by the DALI standard for certain commands.

### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
  - eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
- **nScene**: Scene number (0…15).

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102Reset**

The function block resets all variables of the DALI control gear to their default values.

Control gears can take up to 300 ms to execute the DALI command. Only then are further DALI commands executed by the respective DALI control gear.

---

The function block sends the DALI command RESET twice, as required by the DALI standard for certain commands.

---

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart:** Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress:** Address of a DALI control gear or a DALI group.
- **eAddressType:** Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

---

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

- **bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### FB_DALI102ResetMemoryBank

The values of the selected memory bank of the DALI control gear are set to their respective default values by the function block (see Memory banks [9]).

During the reset, the DALI control gear does not react to other commands. The function block also resets the values that are write-protected by the lock byte.

The DALI commands ENABLE WRITE MEMORY, WRITE MEMORY LOCATION – NO REPLY and RESET MEMORY BANK are called internally.

- The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

- The function block sends the DALI command RESET MEMORY BANK twice, as foreseen by the DALI standard for certain DALI commands.

#### LVAR_INPUT

```plaintext
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nMemoryBank : BYTE := 2;
```

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

- `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
**nMemoryBank**: Specification of the memory bank. If a 0 is specified, all memory banks are reset (except for memory bank 0, as this is read-only). For all other values, only the respective memory bank is reset.

**VAR OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError         : BOOL;</td>
<td></td>
</tr>
<tr>
<td>ipResultMessage: I_TcMessage;</td>
<td></td>
</tr>
<tr>
<td>bBusy          : BOOL;</td>
<td></td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI102SavePersistentVariables**

The function block immediately triggers the saving of variables in the persistent memory by the DALI command SAVE PRESISTENT VARIABLES.

DALI control gears do not save the variables directly in the persistent memory. Instead, the values are copied to the persistent memory in the background. Since this happens over a longer period of time (a few seconds), the values can be lost if the DALI bus is disconnected.

The SAVE PRESISTENT VARIABLES DALI command triggers immediate copying to the persistent memory. In most cases, no communication with the DALI device can take place during copying (up to 350 ms).

The function block sends the DALI command SAVE PRESISTENT VARIABLES twice, as required by the DALI standard for certain commands.

**VAR INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart         : BOOL;</td>
<td></td>
</tr>
<tr>
<td>nAddress       : BYTE;</td>
<td></td>
</tr>
<tr>
<td>eAddressType   : E_DALIAddressType := E_DALIAddressType.Short;</td>
<td></td>
</tr>
<tr>
<td>eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
<td></td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input *nAddress* contains a short address (0...63) or a group address (0...15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

- `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bError</code></td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable <code>ipResultMessage</code>. The output is set to FALSE again as soon as <code>bBusy</code> switches to TRUE.</td>
</tr>
<tr>
<td><code>ipResultMessage</code></td>
<td>Interface pointer (see <code>Error evaluation</code>) that can be used to obtain detailed information about the processing of the function block (see <code>Runtime messages</code>). The interface pointer is valid after <code>bBusy</code> has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td><code>bBusy</code></td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### FB_DALI102SetExtendedFadeTime

This function block writes the values `eExtendedFadeTimeBase` and `eExtendedFadeTimeMultiplier` to the variables `extendedFadeTimeBase` and `extendedFadeTimeMultiplier`.

The Extended Fade Time is calculated from the product of the two variables `extendedFadeTimeBase` and `extendedFadeTimeMultiplier`.

The value can be set between 100 ms and 16 min. With a value of 0 s, the lighting output reaches the required value as quickly as possible.

The `extendedFadeTime` is used in the function blocks `FB_DALI102DirectArcPowerControl` and `FB_DALI102GoToScene`.

The two variables `extendedFadeTimeBase` and `extendedFadeTimeMultiplier` are used for the calculation only if `fadeTime` was set to `E.DALIFadeTime.Disabled` and, if implemented, `fastFadeTime` is equal to `E.DALIFastFadeTime.Disabled`.

The function block changes the DTR0 of all connected DALI control gears.
The function block sends the command SET EXTENDED FADE TIME twice, as foreseen by the DALI standard for certain commands.

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart    : BOOL;</td>
</tr>
<tr>
<td>nAddress  : BYTE;</td>
</tr>
<tr>
<td>eAddressType : E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>eExtendedFadeTimeBase : E_DALIExtendedFadeTimeBase := E_DALIExtendedFadeTimeBase.Base01;</td>
</tr>
<tr>
<td>eExtendedFadeTimeMultiplier : E_DALIExtendedFadeTimeMultiplier := E_DALIExtendedFadeTimeMultiplier.NoFade;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).[547].

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).[549].

   eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

eExtendedFadeTimeBase: Basis for the calculation of the Extended FadeTime (see E_DALIExtendedFadeTimeBase).[552].

eExtendedFadeTimeMultiplier: Time multiplier for the calculation of the Extended FadeTime (see E_DALIExtendedFadeTimeMultiplier).[552].

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError     : BOOL;</td>
</tr>
<tr>
<td>ipResultMessage : I_TcMessage;</td>
</tr>
<tr>
<td>bBusy      : BOOL;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation)[574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages).[561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>
FB_DALI102SetFadeRate

The function block sets the `fadeRate` to the value `eFadeRate`.

The `fadeRate` determines the rate of change, in steps per second, of the output value.

The `fadeRate` is used in the function blocks `FB_DALI102Down` and `FB_DALI102Up`.

The function block changes the DTR0 of all connected DALI control gears.

The function block sends the DALI command SET FADE RATE twice, as required by the DALI standard for certain commands.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).
- **eFadeRate**: Fade rate for the change of the output value (see `E_DALIFadeRate`).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see `Error evaluation`) that can be used to obtain detailed information about the processing of the function block (see `Runtime messages`). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>

FB_DALI102SetFadeTime

The function block sets the fadeTime \([99]\) to the value eFadeTime.

The fadeTime defines the time the current output value takes to reach the required value.

The fadeTime is used in the function blocks FB_DALI102DirectArcPowerControl \([57]\) and FB_DALI102GoToScene \([59]\).

i

The function block changes the DTR0 of all connected DALI control gears.

i

The function block sends the DALI command SET FADE TIME twice, as required by the DALI standard for certain commands.

VAR_INPUT

<table>
<thead>
<tr>
<th>Var</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL; Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE; Address of a DALI device or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short; Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICmdPriority := E_DALICmdPriority.MiddleLow; Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICmdPriority [549]). eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.</td>
</tr>
<tr>
<td>eFadeTime</td>
<td>E_DALIFadeTime := E_DALIFadeTime.Disabled; Fade time for the change of the output value (see E_DALIFadeTime [553]).</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Var</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>
**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>

**FB_DALI102SetMaxLevel**

The function block stores the value `nMaxLevel` in the variable `maxLevel` [98] of the DALI control gear. If the value provided is smaller than `minLevel` then the value is simply set to `minLevel`. If the value 255 (MASK) is specified, `maxLevel` is automatically set to 254.

The function block changes the DTR0 of all connected DALI control gears.

The function block sends the DALI command SET MAX LEVEL twice, as required by the DALI standard for certain commands.

### VAR_INPUT

```plaintext
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nMaxLevel : BYTE := 254;
```

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

*eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
**nMaxLevel**: Maximum allowed output value ($minLevel...254$).

**VAR OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102SetMinLevel**

The function block stores the value $nMinLevel$ in the variable $minLevel$ [98] of the DALI control gear. If a value greater than $maxLevel$ or 255 (MASK) is specified, the value is set to $maxLevel$.

If a value smaller than $physicalMinimum$ is specified, the value is set to $physicalMinimum$.

- The function block changes the DTR0 of all connected DALI control gears.
- The function block sends the DALI command SET MIN LEVEL twice, as required by the DALI standard for certain commands.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nMinLevel</td>
<td>BYTE := 128</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

`eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**nMinLevel**: Minimum allowed output value (physicalMinimum... maxLevel).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102SetPowerOnLevel**

The function block stores the value `nPowerOnLevel` in the variable `powerOnLevel` [98] of the DALI control gear.

The function block changes the DTR0 of all connected DALI control gears.

The function block sends the DALI command SET POWER ON LEVEL twice, as required by the DALI standard for certain commands.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
</tr>
<tr>
<td>nPowerOnLevel</td>
<td>BYTE</td>
</tr>
<tr>
<td>nStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
</tr>
<tr>
<td>nPowerOnLevel</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nPowerOnLevel: Switch-on value (0…255).

VAR_OUTPUT

- bError : BOOL;
- ipResultMessage : I_TcMessage;
- bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102SetScene

The function block stores the value nSceneLevel in the specified scene nScene of the DALI control gear.

If the value is greater than maxLevel and less than 255 (MASK), the value maxLevel is output when the scene is called.

If a value less than minLevel is specified, the value minLevel is output when the scene is called.

Internally, the scenes are stored in the variables scene0 to scene15 [102].

- The function block changes the DTR0 of all connected DALI control gears.
The function block sends the DALI command SET SCENE twice, as required by the DALI standard for certain commands.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nSceneLevel : BYTE := 255;
nScene : BYTE := 0;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

   eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nSceneLevel: Output value of the desired scene (0…255).

nScene: Scene whose output value is to be changed (0…15).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102SetShortAddress
The function block stores the new short address \textit{nNewShortAddress} in the shortAddress variable of the DALI control gear.

The existing short address in the DALI control gear is deleted if a value of 255 is entered at the input.

If you want to give a short address to a device that does not yet have one, you must transmit the command as a broadcast (\textit{eAddressType} = E\_DALIAddressType.Broadcast). This gives all DALI control gears connected to the DALI terminal the short address \textit{nNewShortAddress}.

Alternatively, the function block can be called with Broadcast unaddressed (\textit{eAddressType} = E\_DALIAddressType.BroadcastUnaddr). In this case, only the DALI devices that do not yet have a short address are assigned a short address.

The function block changes the DTR0 of all connected DALI control gears.

The function block sends the DALI command SET SHORT ADDRESS twice, as required by the DALI standard for certain commands.

**VAR\_INPUT**

\begin{verbatim}
  bStart : BOOL;
  nAddress : BYTE;
  eAddressType : E\_DALIAddressType := E\_DALIAddressType.Short;
  eCommandPriority : E\_DALICommandPriority := E\_DALICommandPriority.MiddleLow;
  nNewShortAddress : BYTE;
\end{verbatim}

\textbf{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress}: Address of a DALI device or a DALI group.

\textbf{eAddressType}: Defines whether the input \textit{nAddress} contains a short address (0...63) or a group address (0...15). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \textit{E\_DALIAddressType} [547]).

\textbf{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \textit{E\_DALICommandPriority} [549]).

\textbf{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

\textbf{nNewShortAddress}: New short address (0...63, 255).

**VAR\_OUTPUT**

\begin{verbatim}
  bError : BOOL;
  ipResultMessage : I\_TcMessage;
  bBusy : BOOL;
\end{verbatim}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.

\textbf{ipResultMessage}: Interface pointer (see \textit{Error evaluation} [574]) that can be used to obtain detailed information about the processing of the function block (see \textit{Runtime messages} [561]). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.

\textbf{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102SetSystemFailureLevel

The function block stores the value nSystemFailureLevel in the variable `systemFailureLevel` of the DALI control gear.

- The function block changes the DTR0 of all connected DALI control gears.
- The function block sends the DALI command SET SYSTEM FAILURE LEVEL twice, as required by the DALI standard for certain commands.

**VAR_INPUT**

- **bStart**: BOOL;
- **nAddress**: BYTE;
- **eAddressType**: E_DALIAddressType := E_DALIAddressType.Short;
- **eCommandPriority**: E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
- **nSystemFailureLevel**: BYTE := 254;

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType[547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority[549]).

**eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: BOOL;
- **ipResultMessage**: I_TcMessage;
- **bBusy**: BOOL;

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as **bBusy** switches to TRUE.
ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102StoreActualLevelInDTR0**

The function block writes the current output value of the DALI control gear to DTR0 [102].

The function block sends the DALI command STORE ACTUAL LEVEL IN DTR0 twice, as required by the DALI standard for certain commands.

**VAR_INPUT**

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.1.3 Control commands

FB_DALI102DirectArcPowerControl

The function block switches the DALI control gear to the preset value \( n\text{ArcPowerLevel} \).

If \( n\text{ArcPowerLevel} \) is outside the specified variable \( \text{maxLevel} \) or \( \text{minLevel} \) of the DALI control gear, it is set to the corresponding lowest or highest value. The DALI control gear is switched on by the command if it was previously off.

If \( n\text{ArcPowerLevel} = 0 \), the DALI control gear is switched off. If \( n\text{ArcPowerLevel} = 255 \), the DALI control gear retains its current value.

The speed at which the specified value is to be reached is specified by the variable \( \text{fadeRate} \).

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
</tr>
<tr>
<td>nArcPowerLevel</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input \( n\text{Address} \) contains a short address (0...63) or a group address (0...15). Input \( n\text{Address} \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( \text{E_DALIAddressType} \)).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( \text{E_DALICommandPriority} \)).

\( \text{eCommandPriority} \) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nArcPowerLevel: Control value for the control gear (0...255).

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input \( n\text{Address} \) contains a short address (0...63) or a group address (0...15). Input \( n\text{Address} \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( \text{E_DALIAddressType} \)).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( \text{E_DALICommandPriority} \)).

\( \text{eCommandPriority} \) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nArcPowerLevel: Control value for the control gear (0...255).
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102Down

The output value of the DALI control gear is reduced over the specified period of 200 ms.

The variable fadeRate [98] determines the number of steps by which the output value is changed within the 200 ms.

If the current output value is already at the minLevel [98] value, the output value remains unchanged.

This command does not switch off the DALI control gear.

VAR_INPUT

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart : BOOL;</td>
</tr>
<tr>
<td>nAddress : BYTE;</td>
</tr>
<tr>
<td>eAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError : BOOL;</td>
</tr>
<tr>
<td>ipResultMessage : I_TcMessage;</td>
</tr>
<tr>
<td>bBusy : BOOL;</td>
</tr>
</tbody>
</table>

bError has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102GoToScene

The output of the DALI control gear is set to the value stored in nScene.

The variable fadeTime [99] determines the speed at which the specified value of the scene is to be reached.

If the value of the selected scene in the DALI control gear is 255, the output of the DALI control gear retains its current value.

If the DALI control gear is switched off, this command switches it on.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nScene</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nScene: Scene to be called in the DALI control gear (0…15).
**VAR_OUTPUT**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
<td></td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see *Error evaluation* [574]) that can be used to obtain detailed information about the processing of the function block (see *Runtime messages* [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102Off**

The DALI control gear is switched off immediately.

**VAR_INPUT**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
<td></td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
<td></td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
<td></td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see *E_DALIAddressType* [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see *E_DALICommandPriority* [549]).

- `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

---

**VAR_OUTPUT**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
<td></td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102OnAndStepUp

The output value of the DALI control gear is increased by one step.

If the DALI control gear is off, the command switches it on, and the output value is set to the stored value minLevel [98].

If the output value has already reached the maxLevel [98] value, it retains its current value.

VAR_INPUT

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart : BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress : BYTE</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType : E_DALIAddressType := E_DALIAddressType.Short;</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnadr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError : BOOL</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage : I_TcMessage;</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy : BOOL</td>
<td></td>
</tr>
</tbody>
</table>
**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### FB_DALI102RecallMaxLevel

The output value of the DALI control gear is set to \( \text{maxLevel} \) [98].

If the DALI control gear is switched off, this command switches it on.

#### VAR_INPUT

- **bStart:** Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress:** Address of a DALI control gear or a DALI group.
- **eAddressType:** Defines whether the input \( nAddress \) contains a short address (0…63) or a group address (0…15). Input \( nAddress \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( E\text{\_DALIAddressType} \) [547]).
- **eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( E\text{\_DALICommandPriority} \) [549]).

\[ e\text{CommandPriority} \text{ has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.} \]

#### VAR_OUTPUT

- **bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( ip\text{ResultMessage} \). The output is set to FALSE again as soon as \( b\text{Busy} \) switches to TRUE.
- **ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after \( b\text{Busy} \) has changed from TRUE to FALSE.
- **bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102RecallMinLevel**

The output value of the DALI control gear is set to \texttt{minLevel[\textcopyright 98]} value.

If the DALI control gear is switched off, this command switches it on.

**VAR_INPUT**

- \texttt{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.
- \texttt{nAddress}: Address of a DALI control gear or a DALI group.
- \texttt{eAddressType}: Defines whether the input \texttt{nAddress} contains a short address (0…63) or a group address (0…15). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAddressType[\textcopyright 547]}).
- \texttt{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority[\textcopyright 549]}).

\texttt{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- \texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.
- \texttt{ipResultMessage}: Interface pointer (see \texttt{Error evaluation[\textcopyright 574]}) that can be used to obtain detailed information about the processing of the function block (see \texttt{Runtime messages[\textcopyright 561]}). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.
- \texttt{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
FB_DALI102StepDown

The output value of the DALI control gear is reduced by one step.

If the output of the DALI control gear is already at the minLevel [98] value, the value is not reduced further.

The command does not switch off the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType := E_DALIAddressType.Short;
eCommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage := I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
The output value of the DALI control gear is reduced by one step.

If the output of the DALI control gear is already at the `minLevel` value, it is switched off.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority). `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102StepUp

The output value of the DALI control gear is increased by one step.

If the output value of the DALI control gear is already at the maxLevel value, the value remains unchanged.

The command does not switch on the DALI control gear.

**VAR_INPUT**

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `nAddress`: Address of a DALI control gear or a DALI group.
- `eAddressType`: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).
- `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority). `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- `ipResultMessage`: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- `bBusy`: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
The output value of the DALI control gear is increased over the specified period of 200 ms.

The variable \texttt{fadeRate} determines the number of steps by which the output value is changed within the 200 ms.

If the current output value is already at the \texttt{maxLevel} value, the output value remains unchanged.

This command does not switch on the DALI control gear.

**VAR_INPUT**

\begin{verbatim}
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAдресType := E_DALIAдресType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
\end{verbatim}

\textbf{bStart:} Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress:} Address of a DALI control gear or a DALI group.

\textbf{eAddressType:} Defines whether the input \texttt{nAddress} contains a short address (0…63) or a group address (0…15). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAдресType} [547]).

\textbf{eCommandPriority:} Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority} [549]).

\texttt{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

\begin{verbatim}
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
\end{verbatim}

\textbf{bError:} This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\textbf{ipResultMessage:} Interface pointer (see \texttt{Error evaluation} [574]) that can be used to obtain detailed information about the processing of the function block (see \texttt{Runtime messages} [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\textbf{bBusy:} The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
3.1.2.1.4 Query commands

FB_DALI102QueryActualLevel

The function block reads the current output value of the DALI control gear (see actualLevel [98]).

The current output value is 0 or lies within the value range from minLevel [98] to maxLevel [98].
bError is set to TRUE if the DALI control gear returns the value 255 (MASK). In this case the DALI control gear is in the start-up phase or there is a technical defect.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.
nAddress: Address of a DALI control gear or a DALI group.
eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nActualLevel : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
nActualLevel: Current output value (0, minLevel...maxLevel).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryContentDTR0

The function block reads the content of DTR0 [102] (Data Transfer Register 0) from the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nContentDTR0 : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nContentDTR0: Content of the DTR0 (Data Transfer Register 0) (0…255).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryContentDTR1

The function block reads the content of DTR1 (Data Transfer Register 1) from the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see **E_DALIAddressType**). 
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see **E_DALICommandPriority**). 
  
  eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nContentDTR1**: Content of the DTR1 (Data Transfer Register 1) (0…255).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
The function block reads the content of DTR2 (Data Transfer Register 2) from the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see **E_DALIAddressType**).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see **E_DALICommandPriority**).

> **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see **Error evaluation**) that can be used to obtain detailed information about the processing of the function block (see **Runtime messages**). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nContentDTR2**: Content of the DTR2 (Data Transfer Register 2) (0...255).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryControlGearPresent

The function block indicates whether the required DALI control gear can be reached via its short address.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input \( nAddress \) contains a short address (0…63) or a group address (0…15). Input \( nAddress \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( E\_DALIAddressType \) [\ref{547}]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( E\_DALICommandPriority \) [\ref{549}]).

\[ e\text{CommandPriority} \text{ has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.} \]

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( ipResultMessage \). The output is set to FALSE again as soon as \( bBusy \) switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after \( bBusy \) has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bReady**: The output is set when the queried control gear is available.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryDeviceTypes

The function block returns a list of all device types supported by the DALI control gear.

The DALI commands QUERY DEVICE TYPE and QUERY NEXT DEVICE TYPE are called internally.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]). `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **aDeviceTypes**: List of supported device types (see E_DALI_DeviceType [550]).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryExtendedFadeTime

This function block reads the variables `extendedFadeTimeBase` and `extendedFadeTimeMultiplier` from the DALI control gear. The Extended Fade Time (100 ms to 16 min) can be calculated from the product of the two variables.

The `extendedFadeTime` is used in the function blocks `FB_DALI102DirectArcPowerControl` and `FB_DALI102GoToScene`.

The two variables `extendedFadeTimeBase` and `extendedFadeTimeMultiplier` are used for the calculation only if `fadeTime` was set to `E_DALIFadeTime.Disabled` and, if implemented, `fastFadeTime` is equal to `E_DALIFastFadeTime.Disabled`.

VAR_INPUT

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `nAddress`: Address of a DALI control gear or a DALI group.
- `eAddressType`: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`). `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- `ipResultMessage`: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- `bBusy`: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- `eExtendedFadeTimeBase`: Basis for the calculation of the Extended Fade Time (see `E_DALIExtendedFadeTimeBase`).
eExtendedFadeTimeMultiplier: Time multiplier for the calculation of the Extended Fade Time (see E_DALIExtendedFadeTimeMultiplier [552]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

FB_DALI102QueryFadeTimeFadeRate

The function block reads the variables fadeTime [99] and fadeRate [98] from the DALI control gear.

The fadeTime is used in the function blocks FB_DALI102DirectArcPowerControl [57] and FB_DALI102GoToScene [59], while fadeRate is used with FB_DALI102Up [67] and FB_DALI102Down [58].

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
eFadeTime : E_DALIFadeTime;
eFadeRate : E_DALIFadeRate;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
eFadeTime: Fade time for the change of the output value (see \texttt{E\_DALIFadeTime \cite{553}}).

eFadeRate: Fade rate for the change of the output value (see \texttt{E\_DALIFadeRate \cite{553}}).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>

**FB\_DALI102QueryGroups**

The function block reads the variable \texttt{gearGroups} \cite{102} from the DALI control gear.

Each bit represents one group. Bit 0 (group 0) to bit 15 (group 15). If the bit is set, the DALI control gear belongs to the corresponding group.

The DALI commands QUERY GROUPS 0-7 and QUERY GROUPS 8-15 are called internally.

**VAR\_INPUT**

\begin{verbatim}
VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType : E\_DALIAddressType := E\_DALIAddressType.Short;
eCommandPriority : E\_DALICommandPriority := E\_DALICommandPriority.MiddleLow;
\end{verbatim}

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input \texttt{nAddress} contains a short address (0…63) or a group address (0…15). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E\_DALIAddressType \cite{547}}).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E\_DALICommandPriority \cite{549}}).

\textit{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR\_OUTPUT**

\begin{verbatim}
VAR_OUTPUT
bError : BOOL;
ipResultMessage : I\_TcMessage;
bBusy : BOOL;
nGroups : BYTE;
\end{verbatim}

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

- **ipResultMessage**: Interface pointer (see \texttt{Error evaluation \cite{574}}) that can be used to obtain detailed information about the processing of the function block (see \texttt{Runtime messages \cite{561}}). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
nGroups: Group membership.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102QueryMaxLevel**

The function block reads the variable `maxLevel` from the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).
  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nMaxLevel**: Maximum allowed output value (`minLevel`...254).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryMinLevel

The function block reads the variable minLevel \[98\] from the DALI control gear.

**VAR_INPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType \[547\]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority \[549\]).

* VAR_OUTPUT

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nMinLevel</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation \[574\]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages \[561\]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nMinLevel**: Minimum allowed output value (physicalMinimum... maxLevel).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryPhysicalMinimum

The function block reads the variable `physicalMinimum [102]` from the DALI control gear.

The value is read-only and is fixed by the manufacturer.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType [547]`).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority [549]`).

*Note: `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.*

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nPhysicalMinimum</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nPhysicalMinimum**: Physically smallest possible output value (1…254).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryPowerOnLevel

The function block reads the variable `powerOnLevel` from the DALI control gear.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>: BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>: E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>: E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

\[eCommandPriority\] has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>: I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>nPowerOnLevel</td>
<td>: BYTE;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nPowerOnLevel**: Switch-on value (0…255).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryRandomAddress

The function block reads the variable `randomAddress` from the DALI control gear.

The DALI commands QUERY RANDOM ADDRESS (H), QUERY RANDOM ADDRESS (M) and QUERY RANDOM ADDRESS (L) are called internally.

**VAR_INPUT**

```
VAR_INPUT
  bStart : BOOL;
  nAddress : BYTE;
  eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
  eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
```

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

  - `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

```
VAR_OUTPUT
  bError : BOOL;
  ipResultMessage : I_TcMessage;
  bBusy : BOOL;
  nRandomAddress : UDINT;
```

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nRandomAddress**: Random address (0x00 00 00...0xFF FF FE).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QuerySceneLevel

The function block reads the output value from the DALI control gear for the specified scene \( n\text{Scene} \).

Internally, the scenes are stored in the variables \( \text{scene0} \) to \( \text{scene15} \) (see \( \text{scene0\text{-}scene15} \) [102]).

**VAR_INPUT**

\[
\begin{align*}
b\text{Start} & : \text{BOOL}; \\
n\text{Address} & : \text{BYTE}; \\
e\text{AddressType} & : \text{E_DALIAddressType} := \text{E_DALIAddressType.Short}; \\
e\text{CommandPriority} & : \text{E_DALICommandPriority} := \text{E_DALICommandPriority.MiddleLow}; \\
n\text{Scene} & : \text{BYTE} := 0;
\end{align*}
\]

\( b\text{Start} \): Execution of the DALI commands is triggered via a positive edge at this input.

\( n\text{Address} \): Address of a DALI device or a DALI group.

\( e\text{AddressType} \): Defines whether the input \( n\text{Address} \) contains a short address (0…63) or a group address (0…15). Input \( n\text{Address} \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( \text{E_DALIAddressType} \) [547]).

\( e\text{CommandPriority} \): Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( \text{E_DALICommandPriority} \) [549]).

\( e\text{CommandPriority} \) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

\( n\text{Scene} \): Scene whose output value is to be read (0…15).

**VAR_OUTPUT**

\[
\begin{align*}
b\text{Error} & : \text{BOOL}; \\
\text{ipResultMessage} & : \text{I_TcMessage}; \\
b\text{Busy} & : \text{BOOL}; \\
n\text{SceneLevel} & : \text{BYTE};
\end{align*}
\]

\( b\text{Error} \): This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( \text{ipResultMessage} \). The output is set to FALSE again as soon as \( b\text{Busy} \) switches to TRUE.

\( \text{ipResultMessage} \): Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after \( b\text{Busy} \) has changed from TRUE to FALSE.

\( b\text{Busy} \): The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

\( n\text{SceneLevel} \): Output value of the desired scene (0…255).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102QueryStatus

The function block reads the status of the DALI control gear.

The status contains the eight most important status messages of a DALI control gear. The meaning of the bits is defined as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Status of the DALI control gear (see controlGearFailure [101])</td>
</tr>
<tr>
<td>1</td>
<td>Lamp failure (see lampFailure [101])</td>
</tr>
<tr>
<td>2</td>
<td>Lamp power ON (see lampOn [101])</td>
</tr>
<tr>
<td>3</td>
<td>Limit value error (see limitError [101])</td>
</tr>
<tr>
<td>4</td>
<td>Dimming is active (see fadeRunning [101])</td>
</tr>
<tr>
<td>5</td>
<td>Reset state (see resetState [101])</td>
</tr>
<tr>
<td>6</td>
<td>Short address missing (variable shortAddress [100] is 255 (MASK))</td>
</tr>
<tr>
<td>7</td>
<td>The power supply was enabled (see powerCycleSeen [102])</td>
</tr>
</tbody>
</table>

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
**nStatus:** Status information (see table above).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102QuerySystemFailureLevel**

The function block reads the variable `systemFailureLevel` \[ 98 \] from the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` \[ 547 \]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` \[ 549 \]).

  - `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation \[ 574 \]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages \[ 561 \]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nSystemFailureLevel**: Output value in the case of a system failure (0…255).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
**FB_DALI102ReadMemoryLocation**

One byte is read from the memory bank of the DALI control gear. The memory bank is specified by the parameter `nMemoryBank` and the address within the memory bank by the parameter `nOffset`.

The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

**VAR_INPUT**

- `bStart` : BOOL; Execution of the DALI commands is triggered via a positive edge at this input.
- `nAddress` : BYTE; Address of a DALI device or a DALI group.
- `eAddressType` : E_DALIAddressType := E_DALIAddressType.Short; Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- `eCommandPriority` : E_DALICommandPriority := E_DALICommandPriority.MiddleLow; Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
- `nMemoryBank` : BYTE; Specifies the memory bank for the read access.
- `nOffset` : BYTE; Address from which a value within the memory bank is to be read.

**VAR_OUTPUT**

- `bError` : BOOL; This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- `ipResultMessage` : I_TcMessage; Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- `bBusy` : BOOL; The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- `nValue` : BYTE; Value read from the memory bank.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3.DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

3.1.2.1.5 Special commands

**FB_DALI102EnableDeviceType**

The function block must be called before an application-specific expansion command is called. This means that only the DALI control gears belonging to the corresponding device type respond.

It is not necessary to use the function block for device type 0. It is also not required if a function block is called that represents an application-related expansion command in the library. The DALI command ENABLE DEVICE TYPE is already executed internally in the corresponding function blocks. There is a corresponding note in the description of the respective function blocks.

**VAR_INPUT**

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

  `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

- `eDeviceType`: Selection of the desired device type (see `E_DALIDeviceType` [550]).

**VAR_OUTPUT**

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

  `ipResultMessage`: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- `bBusy`: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3.DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
FB_DALI102Initialise

The function block prepares the addressing of the DALI control gears.

This function block is required if addressing of the DALI control gears is to be implemented in the PLC. The function block FB_DALI102Addressing [12] can be used to assign the addresses from the PLC.

nParameter is used to define which DALI control gears are taken into account for addressing:

<table>
<thead>
<tr>
<th>Value (binary)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000</td>
<td>All DALI control gears respond.</td>
</tr>
<tr>
<td>0AAA AAAA</td>
<td>DALI control gears with the address 00AA AAAA respond.</td>
</tr>
<tr>
<td>1111 1111</td>
<td>DALI control gears without a short address respond.</td>
</tr>
</tbody>
</table>

The function block sends the DALI command INITIALIZE twice, as required by the DALI standard for certain DALI commands.

VAR_INPUT

bStart : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
nParameter : BYTE := 2#1111_1111;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

ECommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nParameter: Specifies which control gears should react to this command (see table above).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102ProgramShortAddress

The function block stores the value \( nShortAddress \) as a short address in the variable \( \text{shortAddress} \) for all selected DALI control gears.

DALI control gears are selected if the \( \text{randomAddress} \) of the DALI control gear is the same as the \( \text{searchAddress} \) and the DALI command INITIALIZE (see \( \text{FB_DALI102Initialise} \)) was executed previously.

This function block is required if addressing of the DALI control gears is to be implemented in the PLC. The function block \( \text{FB_DALI102Addressing} \) can be used to assign the addresses from the PLC.

VAR_INPUT

- \( bStart \): Execution of the DALI commands is triggered via a positive edge at this input.
- \( eCommandPriority \): Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( \text{E_DALICommandPriority} \)).
- \( nShortAddress \): Short address to be issued (0…63, 255).

VAR_OUTPUT

- \( bError \): This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( \text{ipResultMessage} \). The output is set to FALSE again as soon as \( bBusy \) switches to TRUE.
- \( ipResultMessage \): Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after \( bBusy \) has changed from TRUE to FALSE.
- \( bBusy \): The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
**FB_DALI102Randomise**

The function block generates a new random address for all selected DALI control gears and writes it to the `randomAddress` variable.

DALI control gears are selected if the DALI command INITIALIZE (see FB_DALI102Initialise) was executed previously.

This function block is required if addressing of the DALI control gears is to be implemented in the PLC. The function block FB_DALI102Addressing can be used to assign the addresses from the PLC.

- The function block sends the DALI command RANDOMIZE twice, as required by the DALI standard for certain DALI commands.

**VAR_INPUT**

- `bStart` : BOOL;
- `eCommandPriority` : E_DALICommandPriority := E_DALICommandPriority.Middle;

  - `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
  - `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).

  - `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- `bError` : BOOL;
- `ipResultMessage` : I_TcMessage;
- `bBusy` : BOOL;

  - `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

  - `ipResultMessage`: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

  - `bBusy`: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI102SetDTR0

The function block writes to the DTR0 \[102\] (Data Transfer Register 0) of all DALI control gears.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \[549\]). \[549\]
  
  *eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

- **nDTR0**: Value to be written to the DTR0 (Data Transfer Register 0) (0...255).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \[574\]. \[574\] The output is set to FALSE again as soon as \[561\] switches to TRUE.
  
  *ipResultMessage* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

- **ipResultMessage**: Interface pointer (see \[574\]) that can be used to obtain detailed information about the processing of the function block (see \[561\]). The interface pointer is valid after \[561\] has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102SetDTR1

The function block writes to the DTR1 \[102\] (Data Transfer Register 1) of all DALI control gears.
**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.Middle;</td>
</tr>
<tr>
<td>nDTR1</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

*Information: eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.*

**nDTR1**: Value to be written to the DTR1 (Data Transfer Register 1) (0…255).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipAddressResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipAddressResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102SetDTR2**

The function block writes to the DTR2 [102] (Data Transfer Register 2) of all DALI control gears.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.Middle;</td>
</tr>
<tr>
<td>nDTR2</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

*Information: eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.*
nDTR2: Value to be written to the DTR2 (Data Transfer Register 2) (0…255).

**VAR OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102SetSearchAddress**

The function block stores the value *nSearchAddress* in the *searchAddress* variable for all selected DALI control gears.

DALI control gears are selected if the DALI command INITIALIZE (see FB_DALI102Initialise [87]) was executed previously.

The DALI commands SEARCHADDRH, SEARCHADDRM and SEARCHADDRL are called internally.

This function block is required if addressing of the DALI control gears is to be implemented in the PLC. The function block FB_DALI102Addressing [12] can be used to assign the addresses from the PLC.

**VAR INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.Middle;</td>
</tr>
<tr>
<td>nSearchAddress</td>
<td>UDINT;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**nSearchAddress:** The value to be written to the *searchAddress* variable (0x00 00 00…0xFF FF FF).
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102Terminate

The function block terminates addressing for all DALI control gears. The selection of the DALI control gears to be addressed by the FB_DALI102Initialise [87] function block is also canceled.

This function block is required if addressing of the DALI control gears is to be implemented in the PLC. The function block FB_DALI102Addressing [12] can be used to assign the addresses from the PLC.

VAR_INPUT

bStart : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI102VerifyShortAddress**

For all selected DALI control gears, the function block checks whether the variable `shortAddress` matches the value `nShortAddress`.

DALI control gears are selected if the `randomAddress` of the DALI control gear is the same as the `searchAddress` and the DALI command `INITIALIZE` (see `FB_DALI102Initialize`) was executed previously.

This function block is required if addressing of the DALI control gears is to be implemented in the PLC. The function block `FB_DALI102Addressing` can be used to assign the addresses from the PLC.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.Middle;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

`eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nShortAddress: Short address with which the own short address is compared.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bAnswer</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bAnswer: Is TRUE if `nShortAddress` is equal to its own short address.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI102WriteMemoryLocationNoReply

The value \( nValue \) is written in the memory bank of the DALI control gear. The corresponding memory bank is specified by the input variable \( nMemoryBank \), the address within the memory bank by the input variable \( nOffset \).

The DALI commands ENABLE WRITE MEMORY and WRITE MEMORY LOCATION – NO REPLY are called internally.

- The function block changes the DTR0 and the DTR1 of all connected DALI control gears.
- The function block sends the DALI command ENABLE WRITE MEMORY twice, as foreseen by the DALI standard for certain DALI commands.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nMemoryBank</td>
<td>BYTE</td>
</tr>
<tr>
<td>nOffset</td>
<td>BYTE</td>
</tr>
<tr>
<td>nValue</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input \( nAddress \) contains a short address (0...63) or a group address (0...15). Input \( nAddress \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
- **nMemoryBank**: Specifies the memory bank for the write access.
- **nOffset**: Address within the memory bank selected with the variable \( nMemoryBank \).
- **nValue**: Value to be written to the address within the memory bank.
### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>

### 3.1.2.1.6 Variables

Each DALI control gear has a certain number of variables (parameters). These variables are used to configure the DALI control gear and thus influence its behavior. The values of the variables are stored in the respective DALI control gear.
<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>actualLevel</td>
<td>254</td>
<td>0, minLevel...maxLevel</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>powerOnLevel</td>
<td>254</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>systemFailureLevel</td>
<td>254</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>minLevel</td>
<td>PHM</td>
<td>PHM...maxLevel</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>maxLevel</td>
<td>254</td>
<td>minLevel...254</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>fadeRate</td>
<td>E_DALIFadeRate.N04 5StepsPerSec</td>
<td>E_DALIFadeRate.N03StepsPerSec ... E_DALIFadeRate.N358StepsPerSec</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>fadeTime</td>
<td>E_DALIFadeTime.Disabled</td>
<td>E_DALIFadeTime.Disabled ... E_DALIFadeTime. T90500ms</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>extendedFadeTimeBase</td>
<td>E_DALIExtendedFadeTimeBase.Base01</td>
<td>E_DALIExtendedFadeTimeBase. Base01 ... E_DALIExtendedFadeTimeBase. Base15</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>extendedFadeTimeMultiplier</td>
<td>E_DALIExtendedFadeTimeMultiplier.Disabled</td>
<td>E_DALIExtendedFadeTimeMultiplier.Disabled ... E_DALIExtendedFadeTimeMultiplier.Multiplier1min</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>shortAddress</td>
<td>No change</td>
<td>0...63, 255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>searchAddress</td>
<td>16#FF_FF_FF</td>
<td>16#00_00_00...16#FF_FF_FF</td>
<td>3 bytes</td>
<td></td>
</tr>
<tr>
<td>randomAddress</td>
<td>16#FF_FF_FF</td>
<td>16#00_00_00...16#FF_FF_FF</td>
<td>3 bytes</td>
<td></td>
</tr>
<tr>
<td>controlGearFailure</td>
<td>The value can change after a RESET command.</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>lampFailure</td>
<td>The value can change after a RESET command.</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>lampOn</td>
<td>The value can change after a RESET command.</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>limitError</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>fadeRunning</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>resetState</td>
<td>TRUE</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>powerCycleSeen</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>gearGroups</td>
<td>00 00</td>
<td>16#0000...16#FFFF</td>
<td>2 bytes</td>
<td></td>
</tr>
<tr>
<td>scene0-scene15</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>DTR0</td>
<td>No change</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>DTR1</td>
<td>No change</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>DTR2</td>
<td>No change</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>physicalMinLevel (PHM)</td>
<td>No change</td>
<td>1...255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>Name</td>
<td>Reset value</td>
<td>Scope</td>
<td>Size</td>
<td>Comment</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------</td>
<td>--------------------------------------------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>deviceTypes</td>
<td>No change</td>
<td>E_DALI_DeviceType.DT00FluorescentLamp</td>
<td>N bytes</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>E_DALI_DeviceType.DT52DiagnosticMaintenance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

?: undetermined

**actualLevel**

The variable contains the current output value of the DALI control gear. The value can be read with the function block `FB_DALI102QueryActualLevel` [68].

**powerOnLevel**

When the supply voltage is applied to the DALI control gear, the lamp is controlled with the output value stored in the variable `powerOnLevel`. A prerequisite is that the DALI bus is supplied with power and that the quiescent level is maintained. The setting range of `powerOnLevel` is limited by the variables `minLevel` and `maxLevel`. The value can be queried with the function block `FB_DALI102QueryPowerOnLevel` [80] and specified with the function block `FB_DALI102SetPowerOnLevel` [51].

**systemFailureLevel**

If an error occurs on the DALI bus (open-circuit voltage below the permissible level range for more than 500 ms), the DALI control gear is controlled with the output value from the `systemFailureLevel` variable. If the variable value is 255 (MASK), the output value does not change. The value range is limited by the variables `minLevel` and `maxLevel`. The value can be queried with the function block `FB_DALI102QuerySystemFailureLevel` [84] and specified with the function block `FB_DALI102SetSystemFailureLevel` [55].

**minLevel / maxLevel**

The output value is limited within the DALI control gear by the variables `minLevel` and `maxLevel`. Exceptions are the output values 0 (OFF) and 255 (MASK).

**fadeRate**

The `fadeRate` determines the rate of change (in steps per second) of the output value. This variable affects the commands `FB_DALI102Up` [67] and `FB_DALI102Down` [58].

The following values can be specified via `E_DALIFadeRate` [553]:

98         Version: 1.10         TE1000
fadeTime

The **fadeTime** defines the time the current output value takes to reach the required value. If the lamp is switched on, the preheating and ignition times are not included in the fade time. This variable affects the commands `FB_DALI102DirectArcPowerControl` [57] and `FB_DALI102GoToScene` [59].

The following values can be specified via **E_DALIFadeTime** [553]:

<table>
<thead>
<tr>
<th>E_DALIFadeTime</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>If <strong>fadeTime</strong> is set to <strong>E_DALIFadeTime.Disabled</strong>, the Extended Fade Time is used. It is calculated from the product of <strong>extendedFadeTimeBase</strong> and <strong>extendedFadeTimeMultiplier</strong>. If the Extended Fade Time is to be used, then <strong>fastFadeTime</strong>, if present, must also be set to <strong>E_DALIFastFadeTime.Disabled</strong>.</td>
</tr>
<tr>
<td>T00707ms</td>
<td>0.7 s</td>
</tr>
<tr>
<td>T01000ms</td>
<td>1.0 s</td>
</tr>
<tr>
<td>T01400ms</td>
<td>1.4 s</td>
</tr>
<tr>
<td>T02000ms</td>
<td>2.0 s</td>
</tr>
<tr>
<td>T02800ms</td>
<td>2.8 s</td>
</tr>
<tr>
<td>T04000ms</td>
<td>4.0 s</td>
</tr>
<tr>
<td>T05700ms</td>
<td>5.7 s</td>
</tr>
<tr>
<td>T08000ms</td>
<td>8.0 s</td>
</tr>
<tr>
<td>T11300ms</td>
<td>11.3 s</td>
</tr>
<tr>
<td>T16000ms</td>
<td>16.0 s</td>
</tr>
<tr>
<td>T22600ms</td>
<td>22.6 s</td>
</tr>
<tr>
<td>T32000ms</td>
<td>32.0 s</td>
</tr>
<tr>
<td>T45300ms</td>
<td>45.3 s</td>
</tr>
<tr>
<td>T64000ms</td>
<td>64.0 s</td>
</tr>
<tr>
<td>T90500ms</td>
<td>90.5 s</td>
</tr>
</tbody>
</table>

**extendedFadeTimeBase**

This variable is the basic value for the calculation of the Extended Fade Time (0 ms to 16 min). The Extended Fade Time is used if Variables [99] was set to **E_DALIFadeTime.Disabled** and, if present, Variables [177] is equal to **E_DALIFastFadeTime.Disabled**.

---

<table>
<thead>
<tr>
<th>E_DALIFadeRate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>N358StepsPerSec</td>
<td>358 steps/s</td>
</tr>
<tr>
<td>N253StepsPerSec</td>
<td>253 steps/s</td>
</tr>
<tr>
<td>N179StepsPerSec</td>
<td>179 steps/s</td>
</tr>
<tr>
<td>N127StepsPerSec</td>
<td>127 steps/s</td>
</tr>
<tr>
<td>N089StepsPerSec</td>
<td>89.4 steps/s</td>
</tr>
<tr>
<td>N063StepsPerSec</td>
<td>63.3 steps/s</td>
</tr>
<tr>
<td>N045StepsPerSec</td>
<td>44.7 steps/s</td>
</tr>
<tr>
<td>N032StepsPerSec</td>
<td>31.6 steps/s</td>
</tr>
<tr>
<td>N022StepsPerSec</td>
<td>22.4 steps/s</td>
</tr>
<tr>
<td>N016StepsPerSec</td>
<td>15.8 steps/s</td>
</tr>
<tr>
<td>N011StepsPerSec</td>
<td>11.2 steps/s</td>
</tr>
<tr>
<td>N008StepsPerSec</td>
<td>7.9 steps/s</td>
</tr>
<tr>
<td>N006StepsPerSec</td>
<td>5.6 steps/s</td>
</tr>
<tr>
<td>N004StepsPerSec</td>
<td>4.0 steps/s</td>
</tr>
<tr>
<td>N003StepsPerSec</td>
<td>2.8 steps/s</td>
</tr>
</tbody>
</table>
The variable is required in the commands FB_DALI102SetExtendedFadeTime [45] and FB_DALI102QueryExtendedFadeTime [74].

The Extended Fade Time is calculated as follows:

\[
\text{Extended Fade Time} = \text{extendedFadeTimeBase} \times \text{extendedFadeTimeMultiplier}
\]

The following basic values are available:

<table>
<thead>
<tr>
<th>E.DALIExtendeFadeTimeBase</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base01</td>
<td>1</td>
</tr>
<tr>
<td>Base02</td>
<td>2</td>
</tr>
<tr>
<td>Base03</td>
<td>3</td>
</tr>
<tr>
<td>Base04</td>
<td>4</td>
</tr>
<tr>
<td>Base05</td>
<td>5</td>
</tr>
<tr>
<td>Base06</td>
<td>6</td>
</tr>
<tr>
<td>Base07</td>
<td>7</td>
</tr>
<tr>
<td>Base08</td>
<td>8</td>
</tr>
<tr>
<td>Base09</td>
<td>9</td>
</tr>
<tr>
<td>Base10</td>
<td>10</td>
</tr>
<tr>
<td>Base11</td>
<td>11</td>
</tr>
<tr>
<td>Base12</td>
<td>12</td>
</tr>
<tr>
<td>Base13</td>
<td>13</td>
</tr>
<tr>
<td>Base14</td>
<td>14</td>
</tr>
<tr>
<td>Base15</td>
<td>15</td>
</tr>
<tr>
<td>Base16</td>
<td>16</td>
</tr>
</tbody>
</table>

**extendedFadeTimeMultiplier**

This variable is the multiplier for the calculation of the Extended Fade Time (0 ms to 16 min). The Extended Fade Time is used if Variables [99] was set to E.DALIFadeTime.Disabled and, if present, Variables [177] is equal to E.DALIFastFadeTime.Disabled.

If extendedFadeTimeMultiplier assumes the value E.DALIExtendeFadeTimeMultiplier.Disabled, the desired illuminance is reached as soon as possible.

The variable is required in the commands FB_DALI102SetExtendedFadeTime [45] and FB_DALI102QueryExtendedFadeTime [74].

The Extended Fade Time is calculated as follows:

\[
\text{Extended Fade Time} = \text{extendedFadeTimeBase} \times \text{extendedFadeTimeMultiplier}
\]

The following multiplication values are available:

<table>
<thead>
<tr>
<th>E.DALIExtendeFadeTimeMultiplier</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>Fade Time is not used</td>
</tr>
<tr>
<td>Multiplier100ms</td>
<td>x 100 ms</td>
</tr>
<tr>
<td>Multiplier1s</td>
<td>x 1 s</td>
</tr>
<tr>
<td>Multiplier10s</td>
<td>x 10 s</td>
</tr>
<tr>
<td>Multiplier1min</td>
<td>x 1 min</td>
</tr>
</tbody>
</table>

**shortAddress**

The short address is stored in this variable. A valid short address lies in the range between 0 and 63. With a value of 255, the short address is considered deleted. The short address is set with the function block FB_DALI102SetShortAddress [53].
The search address is only required for the assignment of short addresses.

The random address, also known as the long address, is only required when short addresses are being assigned. The function block `FB_DALI102QueryRandomAddress` can be used to read out the 3 bytes of the random address.

The variable indicates whether the DALI control gear operates as intended. In the event of an error, the bit is set to TRUE after 30 s at the latest. The bit is reset once the device works properly again. The variable can be read out with the aid of the function block `FB_DALI102QueryStatus`.

The variable indicates whether the lamp is working correctly. In the event of an error, the bit is set to TRUE after 30 s at the latest. The fault may mean that the connection is faulty or that the lamp is defective. The variable can be read out with the aid of the function block `FB_DALI102QueryStatus`.

The variable indicates whether the lamp is switched on. This bit is set to FALSE when the lamp is off, during the startup phase and in the event of a total lamp failure. The variable can be read out with the aid of the function block `FB_DALI102QueryStatus`.

The variable indicates whether the desired output value `targetLevel` lies within the limits of `minLevel` and `maxLevel`, or whether the `targetLevel` was changed by the values `minLevel` or `maxLevel`. The bit is set to TRUE if one of the described cases occurs. The variable can be read out with the aid of the function block `FB_DALI102QueryStatus`.

This variable indicates whether dimming is active. The bit is TRUE when dimming starts (after the start phase of the light) until the expiry of `fadeTime`. The variable can be read out with the aid of the function block `FB_DALI102QueryStatus`.

The variable indicates whether all device and instance variables have been set to their reset value (see table above). In this case, the bit is TRUE. Variables that show no change in the Reset Value column are not taken into account. The variable can be read out with the aid of the function block `FB_DALI102QueryStatus`.
powerCycleSeen

The variable indicates whether the DALI control gear has been supplied with power without a command having been executed to change the output value.

In this case, the bit is TRUE.

The variable can be read out with the aid of the function block FB_DALI102QueryStatus [83].

gearGroups

The variable indicates whether a particular DALI control gear is assigned to groups 0 to 15. If the bit is set, the control gear belongs to the corresponding group.

The group allocation can be read out with the aid of the function block FB_DALI102QueryGroups [76].

scene0-scene15

Each DALI control gear can store output values for 16 different scenes. There is one output value per scene. If the command for calling a scene FB_DALI102GoToScene [59] is called for a device, a group or all devices (broadcast), the output value of the corresponding DALI control gear is set to the stored value. The output is limited by the values of maxLevel, minLevel and physicalMinLevel.

DTR0-DTR2

Data Transfer Registers 0 to 2 serve as buffer for the execution of various DALI commands.

The DTRs are used by different function blocks. The description of the respective function block indicates which DTRs are used.

Examples for the DTR0 are the query command FB_DALI102QueryContentDTR0 [69] and the command for writing FB_DALI102SetDTR0 [90].

physicalMinLevel

The physically smallest possible output value is stored by the manufacturer in the variable physicalMinLevel. The value can be read with the function block FB_DALI102QueryPhysicalMinimum [79].

deviceTypes

The variable contains the respective device types of a DALI control gear.

The device types (see E_DALIDeviceType [550]) of a DALI control gear can be read using the function block FB_DALI102QueryDeviceTypes [73].

3.1.2.2 Part 103 (control devices)

3.1.2.2.1 Configuration commands

FB_DALI103DisableApplicationController

```
| bStart | BOOL |
| Address | BYTE |
| EDALIAAddressType | E_DALICOMMANDPriority |
| TcEventLogger | I_TdMessage |
| bError | BOOL |
| ResultMessage | BOOL |
| bBusy | BOOL |
```
The function block deactivates the application controller (see `applicationActive` [157]).

The application controller can be enabled with the function block `FB_DALI103EnableApplicationController` [106].

The function block sends the DALI command DISABLE APPLICATION CONTROLLER twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bStart</code></td>
<td>BOOL</td>
</tr>
<tr>
<td><code>nAddress</code></td>
<td>BYTE</td>
</tr>
<tr>
<td><code>eAddressType</code></td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td><code>eCommandPriority</code></td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control device or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bError</code></td>
<td>BOOL</td>
</tr>
<tr>
<td><code>ipResultMessage</code></td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td><code>bBusy</code></td>
<td>BOOL</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI103DisableInstance

The function block deactivates the selected instance of the DALI control device (see instanceActive [159]).

The FB_DALI103EnableInstance [107] function block can be used to enable the instance.

The function block sends the DALI command DISABLE INSTANCE twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after \( b\text{Busy} \) has changed from TRUE to FALSE.

\( b\text{Busy} \): The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103DisablePowerCycleNotification

The function block locks the event Power Cycle Notification.

The event can be released with the function block FB_DALI103EnablePowerCycleNotification [108].

The function block sends the DALI command DISABLE POWER CYCLE NOTIFICATION twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

\[ \begin{align*}
\text{bStart} &: \text{BOOL}; \\
\text{nAddress} &: \text{BYTE}; \\
\text{eAddressType} &: \text{E_DALIAddressType} := \text{E_DALIAddressType.Short}; \\
\text{nInstanceAddress} &: \text{BYTE} := 0; \\
\text{eInstanceAddressType} &: \text{E_DALIInstanceAddressType} := \text{E_DALIInstanceAddressType.InstanceNumber}; \\
\text{eCommandPriority} &: \text{E_DALICommandPriority} := \text{E_DALICommandPriority.MiddleLow};
\end{align*} \]

\( \text{bStart} \): Execution of the DALI commands is triggered via a positive edge at this input.

\( \text{nAddress} \): Address of a DALI device or a DALI group.

\( \text{eAddressType} \): Defines whether the input \( \text{nAddress} \) contains a short address (0…63) or a group address (0…31). Input \( \text{nAddress} \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

\( \text{nInstanceAddress} \): Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

\( \text{eInstanceAddressType} \): Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

\( \text{eCommandPriority} \): Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103EnableApplicationController

The function block activates the application controller (see applicationActive [157]).

The application controller can be disabled with the function block FB_DALI103DisableApplicationController [102].

The function block sends the DALI command ENABLE APPLICATION CONTROLLER twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType :: E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority :: E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103EnableInstance

The function block activates the selected instance of the DALI control device (see instanceActive [159]).

The function block FB_DALI103DisableInstance [104] can be used to disable the instance.

The function block sends the DALI command ENABLE INSTANCE twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.
eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103EnablePowerCycleNotification

The function block releases the event Power Cycle Notification.

The function block FB_DALI103DisablePowerCycleNotification [105] can be used to lock the event.

The function block sends the DALI command ENABLE POWER CYCLE NOTIFICATION twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.
**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td></td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`

**nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see `E_DALIInstanceAddressType`).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

---

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see `Error evaluation`) that can be used to obtain detailed information about the processing of the function block (see `Runtime messages`). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103IdentifyDevice**

The function block starts the identification routine for a DALI control device.

The routine takes approx. 10 s and ends automatically. The exact timing of the identification routine depends on the manufacturer of the control device.
The function block sends the DALI command IDENTIFY DEVICE twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control device or a DALI group.

**eAddressType**: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103Reset**

<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>
The function block resets all device and instance variables of the DALI control device to their default values (see \texttt{resetState} \link{resetState}).

DALI control devices can take up to 300 ms to execute the DALI command. Only then are further DALI commands executed by the respective control device.

\begin{itemize}
  \item The function block sends the DALI command \texttt{RESET} twice, as required by the DALI standard for certain commands.
  \item The function block cannot be used when using the KL6811.
\end{itemize}

\section*{VAR\_INPUT}

\begin{verbatim}
VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
\end{verbatim}

\begin{itemize}
  \item \textbf{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.
  \item \textbf{nAddress}: Address of a DALI control device or a DALI group.
  \item \textbf{eAddressType}: Defines whether the input \texttt{nAddress} contains a short address (0…63) or a group address (0…31). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAddressType} \link{E_DALIAddressType}).
  \item \textbf{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority} \link{E_DALICommandPriority}).
\end{itemize}

The command priorities are supported by the KL6821 from the firmware version BD.

\section*{VAR\_OUTPUT}

\begin{verbatim}
VAR_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
\end{verbatim}

\begin{itemize}
  \item \textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.
  \item \textbf{ipResultMessage}: Interface pointer (see \textit{Error evaluation} \link{Error evaluation}) that can be used to obtain detailed information about the processing of the function block (see \textit{Runtime messages} \link{Runtime messages}). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.
  \item \textbf{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
\end{itemize}

\section*{Requirements}

\begin{table}[H]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Development environment} & \textbf{required PLC library} \\
\hline
TwinCAT from v3.1.4022.29 & Tc3\_DALI from v3.1.4.0 \\
\hline
\end{tabular}
\end{table}
FB_DALI103ResetMemoryBank

The values of the selected memory bank of the DALI control device are set to their respective default values by the function block (see Memory banks [\textsuperscript{9}]).

During the reset, the DALI control device does not react to other commands.

The function block also resets the values that are write-protected by the lock byte.

The DALI commands ENABLE WRITE MEMORY, WRITE MEMORY LOCATION – NO REPLY and RESET MEMORY BANK are called internally.

- The function block changes the DTR0 and the DTR1 of all connected DALI control devices.
- The function block sends the DALI command RESET MEMORY BANK twice, as foreseen by the DALI standard for certain DALI commands.
- The function block cannot be used when using the KL6811.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control device or a DALI group.
- **eAddressType**: Defines whether the input \textit{nAddress} contains a short address (0...63) or a group address (0...15). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \textit{E_DALIAddressType} [\textsuperscript{547}]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \textit{E_DALICommandPriority} [\textsuperscript{549}]).

The command priorities are supported by the KL6821 from the firmware version BD.

- **nMemoryBank**: Specification of the memory bank. If a 0 is specified, all memory banks are reset (except for memory bank 0, as this is read-only). For all other values, only the respective memory bank is reset.

**VAR_OUTPUT**

- **bError**: Indicates an error.
- **ipResultMessage**: Result message.
- **bBusy**: Indicates whether the execution is busy.

---

The function block changes the DTR0 and the DTR1 of all connected DALI control devices.

The function block sends the DALI command RESET MEMORY BANK twice, as foreseen by the DALI standard for certain DALI commands.

The function block cannot be used when using the KL6811.
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI103SavePersistentVariables

The function block immediately triggers the saving of variables in the persistent memory by the DALI command SAVE PERSISTENT VARIABLES.

DALI control devices do not save the variables directly in the persistent memory. Instead, the values are copied to the persistent memory in the background. Since this happens over a longer period of time (a few seconds), the values can be lost if the DALI bus is disconnected.

The SAVE PERSISTENT VARIABLES DALI command triggers immediate copying to the persistent memory. In most cases, no communication with the DALI device can take place during copying (up to 350 ms).

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>: BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>: TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>: BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103SetEventFilter**

The function block sets the event filter (see eventFilter [159]) for the corresponding instance of the DALI control device.

Each bit of the input variable `nEventFilter` represents an event. If the bit is set, the associated event is enabled.

The meanings of the individual bits for the respective instance types can be found here:

- Part 301 (push buttons) – introduction [288]
- Part 302 (input devices, absolute input devices) - introduction [305]
- Part 303 (occupancy sensors) - introduction [314]
- Part 304 (brightness sensors) - introduction [328]

The function block changes the DTR0, DTR1 and DTR2 of all connected DALI control devices.

The function block sends the DALI command SET EVENT FILTER twice, as required by the DALI standard for certain commands.
The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Var</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI device or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>Instance number within a DALI control device. A DALI control device can support multiple instance numbers.</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).</td>
</tr>
<tr>
<td>nEventFilter</td>
<td>Each bit represents an event to be enabled or disabled.</td>
</tr>
</tbody>
</table>

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Var</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI103SetEventPriority

This function block sets the event priority (see eventPriority\[160\]) for the corresponding instance of the DALI control device.

With DALI control devices the event priority affects all Input Notification Events that the respective instance of the DALI control device sends. With DALI control devices, the event priority can assume values from E_DALIEventPriority.Low...E_DALIEventPriority.MiddleHigh.

In most cases the default value should be used.

The function block changes the DTR0 of all connected DALI control devices.

The function block sends the DALI command SET EVENT PRIORITY twice, as foreseen by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

\begin{verbatim}
VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
eEventPriority : E_DALIEventPriority := E_DALIEventPriority.Middle;
\end{verbatim}

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType\[547\]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType\[554\]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority\[549\]).

The command priorities are supported by the KL6821 from the firmware version BD.
**eEventPriority:** Priority (low, middle low, middle, middle high) with which the Input Notification Events are sent from the instance of the DALI control device (see `E_DALIEventPriority` [551]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

- **bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

- **ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- **bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI103SetEventScheme**

The function block determines the event scheme (see `eventScheme` [160]) for the events of the respective DALI control device instance.

- The function block changes the DTR0 of all connected DALI control devices.
- The function block sends the DALI command SET EVENT SCHEME twice, as required by the DALI standard for certain commands.
- The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType_InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>eEventScheme</td>
<td>E_DALIEventScheme := E_DALIEventScheme.DeviceInstance;</td>
</tr>
</tbody>
</table>

- **bStart:** Execution of the DALI commands is triggered via a positive edge at this input.
nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

eEventScheme: Event scheme for the events (see E_DALIEventScheme [551]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103SetOperatingMode

The function block sets the operating mode of the DALI device (see operatingMode [157]).

The function block changes the DTR0 of all connected DALI control devices.
The function block sends the DALI command SET OPERATING MODE twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nOperatingMode</td>
<td>BYTE := 0;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
The function block stores the new short address at `nNewShortAddress` in the DALI control device (see `shortAddress`[156]).

The existing short address in the DALI control device is deleted if a value of 255 is specified at the input.

If you want to give a short address to a device that does not yet have one, you must transmit the command as a broadcast (`eAddressType = E_DALIAddressType.Broadcast`). This gives all control devices connected to the DALI terminal the short address `nNewShortAddress`.

Alternatively, the function block can be called with Broadcast unaddressed (`eAddressType = E_DALIAddressType.BroadcastUnaddr`). In this case, only the devices that do not yet have a short address are assigned a short address.

- The function block changes the DTR0 of all connected DALI control devices.
- The function block sends the DALI command SET SHORT ADDRESS twice, as required by the DALI standard for certain commands.
- The function block cannot be used when using the KL6811.

**VAR_INPUT**

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `nAddress`: Address of a DALI device or a DALI group.
- `eAddressType`: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`[547]).
- `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`[549]). The command priorities are supported by the KL6821 from the firmware version BD.
- `nNewShortAddress`: New short address (0…63, 255).
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103StartQuiescentMode

The function block enables the Quiescent mode [157] for approx. 15 minutes.

In Quiescent mode, no commands and events are sent from the DALI control device.

Quiescent mode can be terminated prematurely with the function block FB_DALI103StopQuiescentMode [122].

The function block sends the DALI command START QUIESCENT MODE twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
The command priorities are supported by the KL6821 from the firmware version BD.

VAR OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>: I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>: BOOL;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103StopQuiescentMode**

The function block interrupts the Quiescent mode [157] prematurely.

Quiescent mode can be started with the function block **FB_DALI103StartQuiescentMode** [121].

The function block sends the DALI command STOP QUIESCENT MODE twice, as required by the DALI standard for certain commands.

The function block cannot be used when using the KL6811.

VAR INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>: BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>: E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>: E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control device or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E.DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3.DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.2 Query commands

FB_DALI103QueryContentDTR0

The function block reads the content of the DTR0 [157] (Data Transfer Register 0) from the DALI control device.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E.DALIAddressType := E.DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E.DALICommandPriority := E.DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E.DALIAddressType [547]).
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nContentDTR0</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE (see Runtime messages [561]).

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nContentDTR0: Content of the DTR0 (Data Transfer Register 0) (0…255).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryContentDTR1

The function block reads the content of the DTR1 (Data Transfer Register 1) from the DALI control device.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nContentDTR1 : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nContentDTR1: Content of the DTR1 (Data Transfer Register 1) (0…255).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryContentDTR2

The function block reads the content of the DTR2 (Data Transfer Register 2) from the DALI control device.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E\_DALICommandPriority}[549]).

The command priorities are supported by the KL6821 from the firmware version BD.

**VAR\_OUTPUT**

\begin{tabular}{l l}
\texttt{bError} & : BOOL; \\
\texttt{ipResultMessage} & : I\_TcMessage; \\
\texttt{bBusy} & : BOOL; \\
\texttt{nContentDTR2} & : BYTE;
\end{tabular}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\textbf{ipResultMessage}: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\textbf{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

\textbf{nContentDTR2}: Content of the DTR2 (Data Transfer Register 2) (0…255).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB\_DALI103QueryDeviceStatus**

The function block reads the status of the DALI control device.

The following status information is provided at output \texttt{nDeviceStatus}.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Status of the DALI control device (see inputDeviceError [158])</td>
</tr>
<tr>
<td>1</td>
<td>Quiescent mode (see quiescentMode [157])</td>
</tr>
<tr>
<td>2</td>
<td>Short address missing (variable shortAddress [156] is 255 (MASK))</td>
</tr>
<tr>
<td>3</td>
<td>Application controller active (see applicationActive [157])</td>
</tr>
<tr>
<td>4</td>
<td>Application controller error (see applicationControllerError [158])</td>
</tr>
<tr>
<td>5</td>
<td>The power supply was enabled (see powerCycleSeen [157])</td>
</tr>
<tr>
<td>6</td>
<td>Reset state (see resetState [158])</td>
</tr>
<tr>
<td>7</td>
<td>Unused (default value is 0)</td>
</tr>
</tbody>
</table>

The function block cannot be used when using the KL6811.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).

i The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nDeviceStatus : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nDeviceStatus: Output value of the Device Status, see table above.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryEventFilter

The function block reads the event filter (see eventFilter) for the selected instance of the DALI control device.

Each bit of the input variable nEventFilter represents an event. If the bit is set, the associated event is enabled.

The meanings of the individual bits for the respective instance types can be found here:
The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
<td>Address of a DALI device or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
<td>Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE</td>
<td>Instance number within a DALI control device. A DALI control device can support multiple instance numbers.</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType</td>
<td>Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation that can be used to obtain detailed information about the processing of the function block (see Runtime messages)). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nEventFilter: Each bit represents a manufacturer-specific event that has been enabled or disabled.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI103QueryEventPriority

This function block reads the event priority (see eventPriority [160]) for the selected instance of the DALI control device.

With DALI control devices the event priority affects all Input Notification Events that the respective instance of the DALI control device sends. With DALI control devices, the event priority can assume values from E_DALIEventPriority.Low...E_DALIEventPriority.MiddleHigh.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.
nAddress: Address of a DALI device or a DALI group.
eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
eEventPriority : E_DALIEventPriority;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**eEventPriority:** Returns the priority (low, middle low, middle, middle high) with which the Input Notification Events are sent from the instance of the DALI control device.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

### FB_DALI103QueryEventScheme

The function block reads the event scheme (see `eventScheme[160]`) for the events of the selected DALI control device instance.

- The function block cannot be used when using the KL6811.

### VAR_INPUT

- **bStart** : BOOL;
- **nAddress** : BYTE;
- **eAddressType** : E_DALIAddressType := E_DALIAddressType.Short;
- **nInstanceAddress** : BYTE := 0;
- **eInstanceAddressType** : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
- **eCommandPriority** : E_DALICmdPriority := E_DALICmdPriority.MiddleLow;

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType[547]`).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see `E_DALIInstanceAddressType[554]`).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICmdPriority[549]`).

- The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
eEventScheme : E_DALIEventScheme;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

eEventScheme: Event scheme for the events (see E_DALIEventScheme [551]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryInputDeviceError

The function block reads more detailed error information of the input devices from the DALI control device (see inputDeviceError [158]).

The meaning depends on the manufacturer of the DALI control device.

- The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nInputDeviceError : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nInputDeviceError: Output of the error information of the input devices of the DALI control device (manufacturer-dependent).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryInputValue

The function block queries the input value (see inputValue [159]) of the selected instance of the DALI control device.

The parameter nResolution specifies how high the resolution (in bits) of the input value is. The function block FB_DALI103QueryResolution [142] can be used to read the resolution.

All bytes of the input value are read and subsequently decoded.

Make sure that the parameter nResolution corresponds to the device's actual resolution, as otherwise decoding errors may occur.

The possible input values for the respective instance types can be found here:

Part 301 (push buttons) – introduction [288]
Part 302 (input devices, absolute input devices) - introduction [305]
Part 303 (occupancy sensors) - introduction [313]
Part 304 (brightness sensors) - introduction [328]

The DALI commands QUERY INPUT VALUE and QUERY INPUT VALUE LATCH are called internally.

The function block cannot be used when using the KL6811.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nResolution : BYTE := 8;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nResolution: Resolution of the input value in bits (1...64).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nInputValue : LWORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nInputValue: The input value of the selected instance of the DALI control device.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI103QueryInstanceEnabled

The function block queries whether the specified instance of the DALI control device is active (see instanceActive [159]).

The FB_DALI103EnableInstance [107] function block can be used to enable the instance.

The function block FB_DALI103DisableInstance [104] can be used to disable the instance.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bInstanceEnabled : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**bInstanceEnabled**: Is TRUE if the instance is active.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### FB_DALI103QueryInstanceError

The function block reads more detailed error information for the selected instance from the DALI control device (see `instanceError`).

The meaning of the values depends on the selected instance type.

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

- **nAddress**: Address of a DALI device or a DALI group.

  - **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).

  - **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

  - **eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see `E_DALIInstanceAddressType`).

  - **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

- **bBusy**: The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

**bError** : BOOL;
**ipResultMessage** : I_TcMessage;
**bBusy** : BOOL;
**nInstanceError** : BYTE;

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable **ipResultMessage**. The output is set to FALSE again as soon as **bBusy** switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after **bBusy** has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nInstanceError**: Error information for the selected instance.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103QueryInstanceStatus**

The function block reads the status of the selected instance from the DALI control device.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Status of the instance (see instanceError [160])</td>
</tr>
<tr>
<td>1</td>
<td>Events are enabled for the instance (see instanceActive [159])</td>
</tr>
<tr>
<td>2-7</td>
<td>Unused (default value is 0)</td>
</tr>
</tbody>
</table>

The function block cannot be used when using the KL6811.

VAR_INPUT

**bStart** : BOOL;
**nAddress** : BYTE;
**eAddressType** : E_DALIAddressType := E_DALIAddressType.Short;
**nInstanceAddress** : BYTE := 0;
**eInstanceAddressType** : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
**eCommandPriority** : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input **nAddress** contains a short address (0...63) or a group address (0...31). Input **nAddress** has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see `E_DALIInstanceAddressType` [554]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

> The command priorities are supported by the KL6821 from the firmware version BD.

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bError</strong></td>
<td>BOOL;</td>
</tr>
<tr>
<td><strong>ipResultMessage</strong></td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td><strong>bBusy</strong></td>
<td>BOOL;</td>
</tr>
<tr>
<td><strong>nInstanceStatus</strong></td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see `Error evaluation` [574]) that can be used to obtain detailed information about the processing of the function block (see `Runtime messages` [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nInstanceStatus:** Status of the selected instance.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103QueryInstanceType**

The function block reads the instance type (see `instanceType` [159]) for the selected instance from the DALI control device.

> The function block cannot be used when using the KL6811.

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bStart</strong></td>
<td>BOOL;</td>
</tr>
<tr>
<td><strong>nAddress</strong></td>
<td>BYTE;</td>
</tr>
<tr>
<td><strong>eAddressType</strong></td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td><strong>nInstanceAddress</strong></td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td><strong>eInstanceAddressType</strong></td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td><strong>eCommandPriority</strong></td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>
**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**nlnInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>eInstanceType</td>
<td>E_DALIInstanceType;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**eInstanceType**: Output of the instance type (see instanceType [159]).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

### FB_DALI103QueryNumberOfInstances

The function block reads the number of existing instances from the DALI control device (see numberOfInstances [157]).

The function block cannot be used when using the KL6811.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nNumberOfInstances : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nNumberOfInstances: Number of instances in the DALI control device.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryOperatingMode

The function block reads the variable operatingMode [157] from the DALI control device.

The meaning of this variable is defined by the manufacturer of the DALI control device.

The function block cannot be used when using the KL6811.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

i

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nOperatingMode : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nOperatingMode: Output of the variable operatingMode [157].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryPowerCycleNotification

The function block outputs whether the Power Cycle Notification event of a DALI control device is enabled.

The function block FB_DALI103EnablePowerCycleNotification [108] can be used to enable the event.

The function block FB_DALI103DisablePowerCycleNotification [105] can be used to disable the event.
The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bPowerCycleNotification : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bPowerCycleNotification: Is TRUE if the Power Cycle Notification event is enabled.

Requirements

Development environment   required PLC library
TwinCAT from v3.1.4022.29   Tc3_DALI from v3.1.4.0

FB_DALI103QueryRandomAddress

The function block reads the randomAddress [156] variable from the DALI control device.

The DALI commands QUERY RANDOM ADDRESS (H), QUERY RANDOM ADDRESS (M) and QUERY RANDOM ADDRESS (L) are called internally.
The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAccessType := E_DALIAccessType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAccessType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nRandomAddress : UDWORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nRandomAddress: Random address (0x00 00 00...0xFF FF FE).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103QueryResolution

The function block reads the resolution of the input value for the selected instance from the DALI control device (see resolution [159]).
The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nResolution : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nResolution: Number of bits with which the input value is resolved.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI103ReadMemoryLocation

The value nValue is read from the memory bank of the DALI controller. The corresponding memory bank is specified by the input variable nMemoryBank, the address within the memory bank by the input variable nOffset.

The function block changes the DTR0 and the DTR1 of all connected DALI control devices.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

- bStart : BOOL;
- nAddress : BYTE;
- eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
- eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
- nMemoryBank : BYTE;
- nOffset : BYTE;

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- The command priorities are supported by the KL6821 from the firmware version BD.

**nMemoryBank**: Specifies the memory bank for the read access.

**nOffset**: Address from which a value within the memory bank is to be read.

**VAR_OUTPUT**

- bError : BOOL;
- ipResultMessage : I_TcMessage;
- bBusy : BOOL;
- nValue : BYTE;

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nValue: Value read from the memory bank.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

3.1.2.2.3 Special commands

FB_DALI103Initialise

The function block prepares the addressing of the DALI control devices.

This function block is required if addressing of the DALI control devices is to be implemented in the PLC. The function block FB_DALI103Addressing [32] can be used to assign the addresses from the PLC.

nParameter is used to define which DALI control devices are taken into account for addressing:

<table>
<thead>
<tr>
<th>Value (binary)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111 1111</td>
<td>DALI control devices without a short address respond.</td>
</tr>
<tr>
<td>00AA AAAA</td>
<td>DALI control devices with the address 00AA AAAA respond.</td>
</tr>
<tr>
<td>1111 1111</td>
<td>All DALI control devices respond.</td>
</tr>
</tbody>
</table>

The function block sends the DALI command INITIALIZE twice, as required by the DALI standard for certain DALI commands.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
eCommandPriority : EDALICommandPriority := EDALICommandPriority.Middle;
nParameter : BYTE := 2#0111 1111;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see EDALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nParameter: Specifies which control units are to respond to this command (see table above).
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103ProgramShortAddress

The function block stores the value nShortAddress as short address in the variable shortAddress [156] for all selected DALI control devices.

DALI control devices are selected if the randomAddress of the DALI control device is the same as the searchAddress and the DALI command INITIALISE (see FB_DALI103Initialise [145]) was executed previously.

This function block is required if addressing of the DALI control devices is to be implemented in the PLC. The function block FB_DALI103Addressing [32] can be used to assign the addresses from the PLC.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
nShortAddress : BYTE;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nShortAddress: Short address to be issued (0…63, 255).
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103Randomise

The function block generates a new random address for all selected DALI control devices and writes it to the randomAddress variable. DALI control devices are selected if the DALI command INITIALISE (see FB_DALI103Initialise [145]) was executed previously.

This function block is required if addressing of the DALI control devices is to be implemented in the PLC. The function block FB_DALI103Addressing [32] can be used to assign the addresses from the PLC.

- The function block sends the DALI command RANDOMIZE twice, as required by the DALI standard for certain DALI commands.

- The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [\$574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [\$561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103SetDTR0**

The function block writes to the DTR0 [\$1571] (Data Transfer Register 0) of all DALI control devices.

The function block cannot be used when using the KL6811.

### VAR_INPUT

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [\$549]).
- `nDTR0`: Value to be written to the DTR0 (Data Transfer Register 0) (0…255).

The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- `ipResultMessage`: Interface pointer (see Error evaluation [\$574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [\$561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**FB_DALI103SetDTR1**

The function block writes to **DTR1** (Data Transfer Register 1) of all DALI control devices.

> The function block cannot be used when using the KL6811.

### VAR_INPUT

**bStart** : BOOL;

**eCommandPriority** : E_DALICommandPriority := E_DALICommandPriority.Middle;

**nDTR1** : BYTE;

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see **E_DALICommandPriority**).

> The command priorities are supported by the KL6821 from the firmware version BD.

**nDTR1**: Value to be written to the DTR1 (Data Transfer Register 1) (0...255).

### VAR_OUTPUT

**bError** : BOOL;

**ipResultMessage** : I_TcMessage;

**bBusy** : BOOL;

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable **ipResultMessage**. The output is set to FALSE again as soon as **bBusy** switches to TRUE.
- **ipResultMessage**: Interface pointer (see **Error evaluation** that can be used to obtain detailed information about the processing of the function block (see **Runtime messages**). The interface pointer is valid after **bBusy** has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
FB_DALI103SetDTR0

The function block writes the DTR0 (Data Transfer Register 0) and the DTR1 (Data Transfer Register 1) of all DALI control devices.

The function block cannot be used when using the KL6811.

VAR_INPUT

| bStart | BOOL; |
| eCommandPriority | E_DALICommandPriority := E_DALICommandPriority.Middle; |
| nDTR0 | BYTE; |
| nDTR1 | BYTE; |

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nDTR0: Value to be written to the DTR0 (Data Transfer Register 0) (0…255).
nDTR1: Value to be written to the DTR1 (Data Transfer Register 1) (0…255).

VAR_OUTPUT

| bError | BOOL; |
| ipResultMessage | I_TcMessage; |
| bBusy | BOOL; |

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>
FB_DALI103SetDTR2

The function block writes to DTR2 [157] (Data Transfer Register 2) of all DALI control devices.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
nDTR2 : BYTE;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nDTR2: Value to be written to the DTR2 (Data Transfer Register 2) (0…255).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103SetSearchAddress

The function block writes to SearchAddress [158] (Search Address) of all DALI control devices.
The function block stores the value \( nSearchAddress \) in the \textit{searchAddress [\texttt{\#\texttt{156}]} \)} variable for all selected DALI control devices.

DALI control devices are selected if the DALI command \textit{INITIALISE} (see \texttt{FB_DALI103 initialised \texttt{[\texttt{\#\texttt{145}]} \)}) was executed previously.

The DALI commands \texttt{SEARCHADDRH, SEARCHADDRM} and \texttt{SEARCHADDRL} are called internally. This function block is required if addressing of the DALI control devices is to be implemented in the PLC. The function block \texttt{FB_DALI103Addressing \texttt{[\texttt{\#\texttt{32}]} \)} can be used to assign the addresses from the PLC.

\begin{itemize}
  \item The function block cannot be used when using the KL6811.
\end{itemize}

\textbf{VAR_INPUT}

\begin{verbatim}
  bStart : BOOL;
  eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.Middle;
  nSearchAddress : UDINT;
\end{verbatim}

\begin{itemize}
  \item \textit{bStart:} Execution of the DALI commands is triggered via a positive edge at this input.
\end{itemize}

\begin{itemize}
  \item \textit{eCommandPriority:} Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority \texttt{[\texttt{\#\texttt{549}]} \)}}).
\end{itemize}

\begin{itemize}
  \item The command priorities are supported by the KL6821 from the firmware version BD.
\end{itemize}

\begin{itemize}
  \item \textit{nSearchAddress:} The value to be written to the \textit{searchAddress} variable (0x00 00 00...0xFF FF FF).
\end{itemize}

\textbf{VAR_OUTPUT}

\begin{verbatim}
  bError : BOOL;
  ipResultMessage : I_TcMessage;
  bBusy : BOOL;
\end{verbatim}

\begin{itemize}
  \item \textit{bError:} This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.
\end{itemize}

\begin{itemize}
  \item \textit{ipResultMessage:} Interface pointer (see \texttt{Error evaluation \texttt{[\texttt{\#\texttt{574}]} \)}} that can be used to obtain detailed information about the processing of the function block (see \texttt{Runtime messages \texttt{[\texttt{\#\texttt{561}]} \)}}). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.
\end{itemize}

\begin{itemize}
  \item \textit{bBusy:} The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
\end{itemize}

\textbf{Requirements}

\begin{itemize}
  \item Development environment \texttt{TwinCAT from v3.1.4022.29}
  \item \texttt{Tc3_DALI from v3.1.4.0}
\end{itemize}

\textbf{FB_DALI103Terminate}

\begin{itemize}
  \item The function block terminates addressing in all DALI control devices. The selection of the devices to be addressed by the function block \texttt{FB_DALI103Initialise \texttt{[\texttt{\#\texttt{145}]} \)}} is also canceled.
\end{itemize}
This function block is required if addressing of the DALI control devices is to be implemented in the PLC. The function block `FB_DALI103Addressing` can be used to assign the addresses from the PLC.

The function block cannot be used when using the KL6811.

### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see `Error evaluation`) that can be used to obtain detailed information about the processing of the function block (see `Runtime messages`). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4.0</td>
<td>Tc3.DALI from v3.1.4</td>
</tr>
</tbody>
</table>

### FB_DALI103VerifyShortAddress

The function block checks whether the variable `shortAddress` matches the value `nShortAddress` for all selected DALI control devices.

DALI control devices are selected if the DALI command `INITIALISE` was executed previously. The DALI commands `SEARCHADDRH`, `SEARCHADDRM` and `SEARCHADDRL` are called internally.

This function block is required if addressing of the DALI control devices is to be implemented in the PLC. The function block `FB_DALI103Addressing` can be used to assign the addresses from the PLC.
The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.Middle;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bAnswer</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

- **bAnswer**: Is TRUE if nShortAddress is equal to its own short address.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

FB_DALI103WriteMemoryLocationNoReply

The value nValue is written in the memory bank of the DALI control device. The corresponding memory bank is specified by the input variable nMemoryBank, the address within the memory bank by the input variable nOffset.
The DALI commands ENABLE WRITE MEMORY and WRITE MEMORY LOCATION – NO REPLY are called internally.

- The function block changes the DTR0 and the DTR1 of all connected DALI control devices.

- The function block sends the DALI command ENABLE WRITE MEMORY twice, as foreseen by the DALI standard for certain DALI commands.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType[547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority[549]).
  - The command priorities are supported by the KL6821 from the firmware version BD.
- **nMemoryBank**: Specifies the memory bank for the write access.
- **nOffset**: Address within the memory bank selected with the variable nMemoryBank.
- **nValue**: Value to be written to the address within the memory bank.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation[574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages[561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>

3.1.2.2.4 Device variables

Each DALI control device has a certain number of device variables (parameters). These device variables are used to configure the DALI control device and thus influence its behavior. The values of the device variables are stored in the respective DALI control device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortAddress</td>
<td>No change</td>
<td>0…63, 255</td>
<td>1 byte</td>
<td>255 (MASK): no address</td>
</tr>
<tr>
<td>searchAddress</td>
<td>16#FF_FF_FF</td>
<td>16#00_00_00…16#FF_FF_FF</td>
<td>3 bytes</td>
<td></td>
</tr>
<tr>
<td>randomAddress</td>
<td>16#FF_FF_FF</td>
<td>16#00_00_00…16#FF_FF_FF</td>
<td>3 bytes</td>
<td></td>
</tr>
<tr>
<td>DTR0 [102]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>DTR1 [102]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>DTR2 [102]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>numberOfInstances</td>
<td>No change</td>
<td>0…32</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>operatingMode</td>
<td>No change</td>
<td>0, 128…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>quiescentMode</td>
<td>DISABLED</td>
<td>ENABLED, DISABLED</td>
<td>1-bit</td>
<td></td>
</tr>
<tr>
<td>applicationActive</td>
<td>No change</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
<tr>
<td>powerCycleSeen</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
<tr>
<td>applicationControllerError</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
<tr>
<td>inputDeviceError</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
<tr>
<td>resetState</td>
<td>TRUE</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
</tbody>
</table>

?: undetermined

**shortAddress**

The short address is stored in this variable. A valid short address lies in the range between 0 and 63. With a value of 255 (MASK), the short address is considered deleted. The short address is set with the function block **FB_DALI103SetShortAddress** [120].

**searchAddress**

The search address is only required for the assignment of short addresses.

**randomAddress**

The random address, also known as the long address, is only required when short addresses are assigned. The function block **FB_DALI103QueryRandomAddress** [141] can be used to read out the 3 bytes of the random address.
DTR0-DTR2
Data Transfer Registers (DTR) 0 to 2 serve as buffer for the execution of various DALI commands.

The DTRs are used by different function blocks. The description of the respective function block indicates which DTRs are used.

Examples for the DTR0 are the query command FB_DALI103QueryContentDTR0 [123] and the command for writing FB_DALI103SetDTR0 [148].

**numOfInstances**
The variable indicates how many instances a DALI control device supports. A device has at least one and a maximum of 32 instances.

The number of instances is queried with the function block FB_DALI103QueryNumberOfInstances [138]. The type of instance is queried with the function block FB_DALI103QueryInstanceType [137].

**operatingMode**
The variable indicates the current operating mode of the DALI control device. The value 0 characterizes the standard operating mode, i.e. the device behaves according to its specification.

The operating modes 0 to 127 are reserved and must not be used.

The operating modes 128 to 255 are vendor-specific.

The function block FB_DALI103QueryOperatingMode [139] can be used to read the operating mode from the DALI control device, the function block FB_DALI103SetOperatingMode [118] can be used to change it.

**quiescentMode**
The variable indicates whether the DALI control device is in quiescent mode. If the variable is TRUE, no commands and events are passed on.

The mode is time-limited to 15 min +/- 1.5 min.

The function block FB_DALI103StartQuiescentMode [121] can be used to start the mode, the function block FB_DALI103StopQuiescentMode [122] can be used to terminate it prematurely.

**applicationActive**
The variable specifies whether the application controller (DALI controller) is active.

According to IEC 62386-103, DALI control devices can contain one or more input devices and optionally a DALI controller. If a DALI controller is contained within a DALI control device, it can independently send DALI commands to other control devices and/or control gears (see DALI [6]).

If the application controller is disabled, it does not send any DALI telegrams other than the Power Cycle Notification.

The application controller can be enabled with the function block FB_DALI103EnableApplicationController [106] and disabled with the function block FB_DALI103DisableApplicationController [102].

**powerCycleSeen**
The variable indicates whether the DALI control device was supplied with power.

In this case, the bit is TRUE.

The variable can be read out with the aid of the function block FB_DALI103QueryDeviceStatus [126].
applicationControllerError
This variable specifies whether an application controller (DALI controller) of a DALI control device has
detected an error.
In this case, the bit is TRUE.
If the DALI control device does not contain an application controller, this variable is always FALSE.
The variable can be read out with the aid of the function block FB_DALI103QueryDeviceStatus [126].

inputDeviceError
The variable indicates whether a DALI control device has detected an error on an input device.
In this case, the bit is TRUE.
The variable can be read out with the aid of the function block FB_DALI103QueryDeviceStatus [126].
The function block FB_DALI103QueryInputDeviceError [131] provides more detailed information about the
present error. The output depends on the manufacturer.

resetState
The variable indicates whether all device and instance variables have been set to their reset value (see table
above).
In this case, the bit is TRUE.
The variable can be read out with the aid of the function block FB_DALI103QueryDeviceStatus [126].

3.1.2.2.5 Instance variables
Each DALI control device has at least one and a maximum of 32 instances (see variable
numberOfInstances). Each instance has a certain number of instance variables (parameters), irrespective of
the other instances. These instance variables are used to configure the respective instance in the DALI
control device and thus influence its behavior. The values of the instance variables are stored in the
respective DALI control device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>instanceActive [159]</td>
<td>No change</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
<tr>
<td>instanceType [159]</td>
<td>No change</td>
<td>0…31</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>resolution [159]</td>
<td>No change</td>
<td>1…255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>inputValue [159]</td>
<td>No change</td>
<td>0…(2^N-1)</td>
<td>N bytes</td>
<td>N bytes (max. 32 bytes)</td>
</tr>
<tr>
<td>eventFilter [159]</td>
<td>16#FF_FF_FF</td>
<td>16#00_00_00…16#FF_FF_FF</td>
<td>3 bytes</td>
<td></td>
</tr>
<tr>
<td>eventScheme [160]</td>
<td>E_DALIEventScheme.Instance</td>
<td>... E_DALIEventScheme.Instance.Group</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>eventPriority [160]</td>
<td>No change</td>
<td>E_DALIEventPriority.Low ... E_DALIEventPriority.MiddleHigh</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>instanceError [160]</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>1-bit</td>
<td></td>
</tr>
</tbody>
</table>

?: undetermined
N: resolution
a: Other values can be defined for individual instance types. Details can be taken from the IEC 62386-3xx standard.

**instanceActive**

The variable indicates whether sending of events has been enabled for the respective instance.

If the variable is FALSE, the instance does not send any events. This variable has no effect on the DALI commands sent to the instance by other DALI controllers. These will be answered even if `instanceActive` is FALSE.

The function block `FB_DALI103EnableInstance` is used to enable sending of events, the function block `FB_DALI103DisableInstance` is used to disable it.

**instanceType**

The variable contains the respective instance type of an instance.

<table>
<thead>
<tr>
<th>Instance type</th>
<th>IEC 62386-</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>103</td>
<td>General application and control devices; not defined further</td>
</tr>
<tr>
<td>1-31</td>
<td>301-331</td>
<td>These IEC 62386-3xx parts describe instance types from 301 to 331</td>
</tr>
</tbody>
</table>

The instance type (see `E_DALIInstanceType`) of an instance can be read using the function block `FB_DALI103QueryInstanceType`.

**resolution**

The variable indicates the resolution of `inputValue`.

The resolution corresponds to the number of bits through which the input value is mapped.

The function block `FB_DALI103QueryResolution` can be used to read the resolution.

**inputValue**

The variable outputs the input value of an instance.

The number of bits in which the input value is mapped can be queried through the variable `resolution`.

The input value can be read out with the function block `FB_DALI103QueryInputValue`.

**eventFilter**

The event filter enables or disables specific events for each instance. The meanings of the individual bits for the respective instance types can be found here:

Part 301 (push buttons) – Introduction

Part 302 (absolute encoder) - Introduction

Part 303 (occupancy sensors) - Introduction

Part 304 (light sensors) - Introduction

The function block `FB_DALI103QueryEventFilter` can be used to read the value of the event filter; the function block `FB_DALI103SetEventFilter` can be used to write to it.
eventScheme

The variable contains the event scheme for the respective instance of a control device.

<table>
<thead>
<tr>
<th>E_DALIEventScheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>Instance addressing with instance type and number.</td>
</tr>
<tr>
<td>Device</td>
<td>Device addressing with short address and instance type.</td>
</tr>
<tr>
<td>DeviceInstance</td>
<td>Device/instance addressing with short address and instance number.</td>
</tr>
<tr>
<td>DeviceGroup</td>
<td>Device group addressing with device group and instance type.</td>
</tr>
<tr>
<td>InstanceGroup</td>
<td>Instance group addressing with instance group and type.</td>
</tr>
</tbody>
</table>

The function block FB_DALI103QueryEventScheme [130] can be used to read the event scheme; the function block FB_DALI103SetEventScheme [117] can be used to write to it.

eventPriority

Different Priorities [8] for sending events can be configured for each instance. The device function blocks use the following default values for event priorities:

<table>
<thead>
<tr>
<th>E_DALIEventPriority</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiddleHigh</td>
<td></td>
</tr>
<tr>
<td>Middle</td>
<td>Part 301 (push button), Part 302 (absolute encoder), Part 303 (occupancy sensor)</td>
</tr>
<tr>
<td>MiddleLow</td>
<td>Part 304 (light sensors)</td>
</tr>
<tr>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

The value of the event priority can be read with the function block FB_DALI103QueryEventPriority [129] and written with the function block FB_DALI103SetEventPriority [116].

instanceError

The variable indicates whether an instance of a control device has detected an error.

In this case, the bit is TRUE.

The variable can be read out with the aid of the function block FB_DALI103QueryInstanceStatus [136].

The function block FB_DALI103QueryInstanceError [135] provides more detailed information about the present error. The output depends on the manufacturer.

3.1.2.3 Part 207 (LED modules)

3.1.2.3.1 Introduction

IEC 62386-207 describes DALI control gears for controlling LED modules.

This part includes functions that go beyond IEC 62386-102 for controlling LED modules.

Functionality

The Tc3_DALI library offers function blocks for setting various parameters:

- Referencing system performance to detect a significant increase or drop.
- Dimming curve (standard or linear)
- Fast Fade Time (see Variables [177])

Furthermore, the library contains function blocks for querying various parameters:
• Device type (see Variables [177])
• Dimming curve (see Variables [178])
• Querying the failure status (see Variables [178])
• Fast Fade Time (see Variables [177] or Variables [177])
• Device-specific features (see Variables [177])
• Checking the load deviation
• Querying the referencing

### 3.1.2.3.2 Configuration commands

**FB_DALI207ReferenceSystemPower**

The DALI control gear measures and stores the performance level of the system, in order to detect load increase and decrease.

Prior to the DALI command REFERENCE SYSTEM POWER, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

The function block sends the DALI command REFERENCE SYSTEM POWER twice, as foreseen by the DALI standard for certain commands.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

*bStart*: Execution of the DALI commands is triggered via a positive edge at this input.

*nAddress*: Address of a DALI control gear or a DALI group.

*eAddressType*: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

*eCommandPriority*: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

*eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207SelectDimmingCurve**

The function block saves the value eDimmingCurve in the variable dimmingCurve [178] of the DALI control gear.

Prior to the DALI command SELECT DIMMING CURVE, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

The function block sends the DALI command SELECT DIMMING CURVE twice, as foreseen by the DALI standard for certain commands.

The function block changes the DTR0 of all connected DALI control gears.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALIServiceCommandPriority := E_DALIServiceCommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>eDimmingCurve</td>
<td>E_DALIDimmingCurve := E_DALIDimmingCurve.Standard;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALIServiceCommandPriority [549]).
eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**eDimmingCurve:** Linear or standard dimming curve (see E_DALIDimmingCurve [551])

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207SetFastFadeTime**

The function block saves the value *eFastFadeTime* in the variable `fastFadeTime [177]` of the DALI control gear.

The *fastFadeTime* is used in the function blocks **FB_DALI102DirectArcPowerControl [57]** and **FB_DALI102GoToScene [59]**.

Prior to the DALI command SET FAST FADE TIME, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application extended commands (see also **FB_DALI102EnableDeviceType [86]**). The function block sends the DALI command SET FAST FADE TIME twice, as required by the DALI standard for certain commands.

The function block changes the DTR0 of all connected DALI control gears.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
eFastFadeTime : E_DALIFastFadeTime := E_DALIFastFadeTime.Disabled;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

eFastFadeTime: Fast fade time for the change in the output value (see E_DALIFastFadeTime [553]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.3.3 Query commands

FB_DALI207QueryControlGearType

The function block reads the variable **controlGearType [177]** from the DALI control gear.
Prior to the DALI command QUERY CONTROL GEAR TYPE, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nControlGearType</td>
<td>Outputs the value of the variable controlGearType (0…255).</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI207QueryDimmingCurve

The function block reads the set dimming curve from the DALI control gear (see E_DALIDimmingCurve [551]).
Prior to the DALI command QUERY DIMMING CURVE, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [\ref{86}]).

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

- **nAddress**: Address of a DALI control gear or a DALI group.

- **eAddressType**: Defines whether the input *nAddress* contains a short address (0...63) or a group address (0...15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [\ref{547}]).

- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [\ref{549}]).

  - *eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

  - **ipResultMessage**: Interface pointer (see error evaluation [\ref{574}]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [\ref{561}]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

- **eDimmingCurve**: Linear or standard dimming curve.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207QueryFailureStatus**

The function block reads the failure status from the DALI control gear (see failureStatus [\ref{178}]).
Prior to the DALI command QUERY FAILURE STATUS, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nFailureStatus</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nFailureStatus**: Returns the failure status (see failureStatus [178]).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207QueryFastFadeTime**

The function block reads the variable fastFadeTime [177] from the DALI control gear.
Programming

The fastFadeTime is used in the function blocks FB_DALI102DirectArcPowerControl [57] and FB_DALI102GoToScene [59].

Prior to the DALI command QUERY FAST FADE TIME, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
eFastFadeTime : E_DALIFastFadeTime;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

eFastFadeTime: Output value of the fastFadeTime parameter (see E_DALIFastFadeTime [553]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI207QueryFeatures

The function block reads the variable features [177] from the DALI control gear.

TE1000
Prior to the DALI command QUERY FEATURES, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nFeatures : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nFeatures: Returns information on the implemented properties (see features [177]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI207QueryLoadDecrease

The function block queries whether a significant load decrease (in comparison with the reference power of the system) has been detected.
Prior to the DALI command QUERY LOAD DECREASE, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see F_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see F_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nLoadDecrease</td>
<td>Detection of a significant load decrease.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207QueryLoadIncrease**

The function block queries whether a significant load increase (in comparison with the reference power of the system) has been detected.
Prior to the DALI command QUERY LOAD INCREASE, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.  

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType [547]`).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority [549]`).  

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bLoadIncrease</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**bLoadIncrease**: Detection of a significant load increase.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207QueryMinFastFadeTime**

The function block reads the variable `minFastFadeTime [177]` from the DALI control gear.
Prior to the DALI command MIN FAST FADE TIME, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType \[\text{\textsection 86}\]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input \textit{nAddress} contains a short address (0...63) or a group address (0...15). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \textit{E_DALIAddressType} \[\text{\textsection 547}\]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \textit{E_DALICommandPriority} \[\text{\textsection 549}\]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>eMinFastFadeTime</td>
<td>E_DALIFastFadeTime;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.

**ipResultMessage:** Interface pointer (see \textit{error evaluation} \[\text{\textsection 574}\]) that can be used to obtain detailed information about the processing of the function block (see \textit{runtime messages} \[\text{\textsection 561}\]). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**eMinFastFadeTime:** Returns the minimum value of the \textit{fastFadeTime} parameter (see \textit{E_DALIFastFadeTime} \[\text{\textsection 553}\]).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207QueryReferenceMeasurementFailed**

The function block queries whether a started reference measurement has failed.
Prior to the DALI command QUERY REFERENCE MEASUREMENT FAILED, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICmdPriority := E_DALICmdPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICmdPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : TcMessage;
bBusy : BOOL;
bReferenceMeasurementFailed : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bReferenceMeasurementFailed: Indicates whether the reference measurement failed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI207QueryReferenceRunning

The function block queries whether a reference measurement of the system power is activated.
Prior to the DALI command QUERY REFERENCE RUNNING, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bReferenceRunning : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bReferenceRunning: Indicates whether the reference measurement of the system performance is performed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI207QueryThermalOverload

The function block queries whether there is a thermal overload with reduction of the luminous flux.
Prior to the DALI command QUERY THERMAL OVERLOAD, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bThermalOverload**: Indicates whether thermal overload is present.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI207QueryThermalShutdown**

The function block queries whether a thermal shutdown has taken place.
Prior to the DALI command QUERY THERMAL SHUTDOWN, the function block sends the DALI command ENABLE DEVICE TYPE 6, as is necessary for application-related expansion commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICmdPriority := E_DALICmdPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICmdPriority [549]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bThermalShutdown</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bThermalShutdown**: Indicates whether a thermal shutdown has taken place.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.3.4 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>minFastFadeTime</td>
<td>No change</td>
<td>E_DALIFastFadeTime.T100ms</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>E_DALIFastFadeTime.T700ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastFadeTime</td>
<td>E_DALIFastFadeTime.Disabled</td>
<td>E_DALIFastFadeTime.Disabled, minFastFadeTime</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>E_DALIFastFadeTime.T700ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>controlGearType</td>
<td>No change</td>
<td>0...255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>features</td>
<td>No change</td>
<td>0...255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>failureStatus</td>
<td>No change</td>
<td>0...255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>dimmingCurve</td>
<td>E_DALIDimmingCurve.Standard</td>
<td>E_DALIDimmingCurve.Standard, E_DALIDimmingCurve.Linear</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

?: undetermined

minFastFadeTime

The minFastFadeTime specifies the shortest fadeTime, within which the output of a DALI control device behaves in accordance with the selected dimming curve. Its value can lie between E_DALIFastFadeTime.T100ms and E_DALIFastFadeTime.T700ms.

fastFadeTime

The fastfadeTime is used instead of the fadeTime if the fadeTime is equal to E_DALIFastFadeTime.Disabled. The fastFadeTime can be programmed to E_DALIFastFadeTime.Disabled or any value in the range between the value in the variable minFastFadeTime and E_DALIFastFadeTime.T700ms. Programming the fastFadeTime to E_DALIFastFadeTime.Disabled means the fastest possible change of the luminous flux.

The fastFadeTime is used in the function blocks FB_DALI102DirectArcPowerControl [57] and FB_DALI102GoToScene [59].

controlGearType

The variable contains information about the device type:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserve</td>
</tr>
<tr>
<td>1</td>
<td>LED module integrated</td>
</tr>
<tr>
<td>2</td>
<td>Reserve</td>
</tr>
<tr>
<td>3</td>
<td>Reserve</td>
</tr>
<tr>
<td>4</td>
<td>Reserve</td>
</tr>
<tr>
<td>5</td>
<td>Reserve</td>
</tr>
<tr>
<td>6</td>
<td>Reserve</td>
</tr>
<tr>
<td>7</td>
<td>Reserve</td>
</tr>
</tbody>
</table>

features

The variable contains information about the implemented optional properties:
### failureStatus

The variable contains information about the failure status:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserve</td>
</tr>
<tr>
<td>1</td>
<td>Reserve</td>
</tr>
<tr>
<td>2</td>
<td>Load decrease</td>
</tr>
<tr>
<td>3</td>
<td>Load increase</td>
</tr>
<tr>
<td>4</td>
<td>Reserve</td>
</tr>
<tr>
<td>5</td>
<td>Thermal shutdown</td>
</tr>
<tr>
<td>6</td>
<td>Thermal overload with reduction of the luminous flux</td>
</tr>
<tr>
<td>7</td>
<td>Reference measurement failed</td>
</tr>
</tbody>
</table>

### dimmingCurve

The variable specifies the type of dimming curve.

<table>
<thead>
<tr>
<th>E_DALIDimmingCurve</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Standard dimming curve</td>
</tr>
<tr>
<td>Linear</td>
<td>Linear dimming curve</td>
</tr>
</tbody>
</table>

### 3.1.2.4 Part 208 (device for switching functions)

#### 3.1.2.4.1 Introduction

#### 3.1.2.4.2 Configuration commands

**FB_DALI208SetDownSwitchOffThreshold**

This function block saves the value `nDownSwitchOffThreshold` in the variable `downSwitchOffThreshold` of the DALI control gear.
The output of the DALI control gear is reset as soon as the falling virtual output value reaches or falls below the threshold value \textit{downSwitchOffThreshold}.

The threshold value is disabled with the value \textit{nDownSwitchOffThreshold} = 255 (MASK).

The DALI command is then only supported by a DALI control gear if bit 3 of \textit{features} \[191\] is set.

Prior to the DALI command SET DOWN SWITCH-OFF THRESHOLD, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also \textit{FB_DALI102EnableDeviceType} \[86\]).

The function block sends the DALI command SET DOWN SWITCH-OFF THRESHOLD twice, as foreseen by the DALI standard for certain commands.

The function block changes the DTR0 of all connected DALI control gears.

\textbf{VAR_INPUT}

\begin{tabular}{|l|}
\hline
\textbf{bStart} & : BOOL; \\
\textbf{nAddress} & : BYTE; \\
\textbf{eAddressType} & : E.DALIAddressType := E.DALIAddressType.Short; \\
\textbf{eCommandPriority} & : E.DALICommandPriority := E.DALICommandPriority.MiddleLow; \\
\textbf{nDownSwitchOffThreshold} & : BYTE := 0; \\
\hline
\end{tabular}

\textbf{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress}: Address of a DALI control gear or a DALI group.

\textbf{eAddressType}: Defines whether the input \textbf{nAddress} contains a short address (0...63) or a group address (0...15). Input \textbf{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \textit{E.DALIAddressType} \[547\]).

\textbf{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \textit{E.DALICommandPriority} \[549\]).

\textit{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

\textbf{nDownSwitchOffThreshold}: Threshold value (0...255) for switching off the output with a falling virtual output value.

\textbf{VAR_OUTPUT}

\begin{tabular}{|l|}
\hline
\textbf{bError} & : BOOL; \\
\textbf{ipResultMessage} & : I_TcMessage; \\
\textbf{bBusy} & : BOOL; \\
\hline
\end{tabular}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textbf{bBusy} switches to TRUE.

\textbf{ipResultMessage}: Interface pointer (see \textit{Error evaluation} \[574\]) that can be used to obtain detailed information about the processing of the function block (see \textit{Runtime messages} \[561\]). The interface pointer is valid after \textbf{bBusy} has changed from TRUE to FALSE.

\textbf{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

FB_DALI208SetDownSwitchOnThreshold

This function block saves the value \( n_{\text{DownSwitchOnThreshold}} \) in the variable \( \text{downSwitchOnThreshold} \) \( \text{[191]} \) of the DALI control gear.

The output of the DALI control gear is reset as soon as the falling virtual output value reaches or falls below the threshold value \( \text{downSwitchOnThreshold} \).

The threshold value is disabled with the value \( n_{\text{DownSwitchOnThreshold}} = 255 \) (MASK).

The DALI command is then only supported by a DALI control gear if bit 3 of \( \text{features} \) \( \text{[191]} \) is set.

- Prior to the DALI command SET DOWN SWITCH-ON THRESHOLD, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType \( \text{[86]} \)).
- The function block sends the DALI command SET DOWN SWITCH-ON THRESHOLD twice, as foreseen by the DALI standard for certain commands.
- The function block changes the DTR0 of all connected DALI control gears.

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b_{\text{Start}} ) : BOOL;</td>
</tr>
<tr>
<td>( n_{\text{Address}} ) : BYTE;</td>
</tr>
<tr>
<td>( e_{\text{AddressType}} ) := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>( e_{\text{CommandPriority}} ) := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>( n_{\text{DownSwitchOnThreshold}} ) : BYTE := 255;</td>
</tr>
</tbody>
</table>

- \( b_{\text{Start}} \): Execution of the DALI commands is triggered via a positive edge at this input.
- \( n_{\text{Address}} \): Address of a DALI control gear or a DALI group.
- \( e_{\text{AddressType}} \): Defines whether the input \( n_{\text{Address}} \) contains a short address (0…63) or a group address (0…15). Input \( n_{\text{Address}} \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType \( \text{[547]} \)).
- \( e_{\text{CommandPriority}} \): Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority \( \text{[549]} \)).
- \( e_{\text{CommandPriority}} \) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
- \( n_{\text{DownSwitchOffThreshold}} \): Threshold value (0…255) for switching on the output with a falling virtual output value.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI208SetUpSwitchOffThreshold**

This function block saves the value nUpSwitchOffThreshold in the variable upSwitchOffThreshold [191] of the DALI control gear.

The output of the DALI control gear is reset as soon as the rising virtual output value reaches or exceeds the threshold value upSwitchOffThreshold.

The threshold value is disabled with the value nUpSwitchOffThreshold = 255 (MASK).

The DALI command is then only supported by a DALI control gear if bit 3 of features [191] is set.

Prior to the DALI command SET UP SWITCH-OFF THRESHOLD, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType [86]).

The function block sends the DALI command SET UP SWITCH-OFF THRESHOLD twice, as foreseen by the DALI standard for certain commands.

The function block changes the DTR0 of all connected DALI control gears.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nUpSwitchOffThreshold : BYTE := 255;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.
eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nUpSwitchOffThreshold: Threshold value (0…255) for switching off the output with a rising virtual output value.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

FB_DALI208SetUpSwitchOnThreshold

This function block saves the value nUpSwitchOnThreshold in the variable upSwitchOnThreshold [191] of the DALI control gear.

The output of the DALI control gear is reset as soon as the rising virtual output value reaches or exceeds the threshold value upSwitchOnThreshold.

The threshold value is disabled with the value nUpSwitchOnThreshold = 255 (MASK).

The DALI command is then only supported by a DALI control gear if bit 3 of features [191] is set.

- Prior to the DALI command SET UP SWITCH-ON THRESHOLD, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType [86]).

- The function block sends the DALI command SET UP SWITCH-ON THRESHOLD twice, as foreseen by the DALI standard for certain commands.
The function block changes the DTR0 of all connected DALI control gears.

**VAR_INPUT**

```plaintext
VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nUpSwitchOnThreshold : BYTE := 1;
```

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**nUpSwitchOnThreshold:** Threshold value (0...255) for switching off the output with a rising virtual output value.

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
```

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**3.1.2.4.3 Query commands**

**FB_DALI208QueryDownSwitchOffThreshold**
The function block reads the variable `downSwitchOffThreshold` from the DALI control gear.

The output of the DALI control gear is reset as soon as the falling virtual output value reaches or falls below the threshold value `downSwitchOffThreshold`.

Prior to the DALI command QUERY DOWN SWITCH-OFF, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType).

**VAR_INPUT**

```
bStart            : BOOL;
nAddress          : BYTE;
eAddressType      : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
```

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).

*Note:* `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

```
bError            : BOOL;
ipResultMessage  : I_TcMessage;
bBusy            : BOOL;
nDownSwitchOffThreshold : BYTE;
```

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nDownSwitchOffThreshold**: Threshold value (0…255) for switching off the output with a falling virtual output value.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI208QueryDownSwitchOnThreshold**
The function block reads the variable \texttt{downSwitchOnThreshold} from the DALI control gear.

The output of the DALI control gear is set as soon as the falling virtual output value reaches or falls below the threshold value \texttt{downSwitchOnThreshold}.

Prior to the DALI command QUERY DOWN SWITCH-ON, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also \texttt{FB_DALI102EnableDeviceType} [86]).

**VAR_INPUT**

- \texttt{bStart} \hspace{1em} BOOL;
- \texttt{nAddress} \hspace{1em} BYTE;
- \texttt{eAddressType} \hspace{1em} \texttt{E_DALIAddressType} := \texttt{E_DALIAddressType.Short};
- \texttt{eCommandPriority} \hspace{1em} \texttt{E_DALICommandPriority} := \texttt{E_DALICommandPriority.MiddleLow};

\textbf{bStart:} Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress:} Address of a DALI control gear or a DALI group.

\textbf{eAddressType:} Defines whether the input \texttt{nAddress} contains a short address (0…63) or a group address (0…15). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAddressType} [547]).

\textbf{eCommandPriority:} Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority} [549]).

\textbf{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- \texttt{bError} \hspace{1em} BOOL;
- \texttt{ipResultMessage} \hspace{1em} \texttt{I_TcMessage};
- \texttt{bBusy} \hspace{1em} BOOL;
- \texttt{nDownSwitchOnThreshold} \hspace{1em} BYTE;

\textbf{bError:} This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\textbf{ipResultMessage:} Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\textbf{bBusy:} The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

\textbf{nDownSwitchOnThreshold:} Threshold value (0…255) for switching on the output with a falling virtual output value.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI208QueryFeatures**
The function block reads the variable features[191] from the DALI control gear.

Prior to the DALI command QUERY FEATURES, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType[86]).

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType[547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority[549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nFeatures : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation[574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages[561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nFeatures: Contains information on the implemented properties (see features[191]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3 DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

FB_DALI208QueryGearType

The function block reads the variable gearType[192] from the DALI control gear.
Prior to the DALI command QUERY GEAR TYPES, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType [» 86]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [» 547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [» 549]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nGearType</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage:** Interface pointer (see error evaluation [» 574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [» 561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nGearType:** Contains information on the properties of the DALI control gear (see gearType [» 192]).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI208QuerySwitchStatus**

This function block reads the variable switchStatus [» 192] from the DALI control gear.
Prior to the DALI command QUERY SWITCH STATUS, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `F_DALIAddressType [547]`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `F_DALICommandPriority [549]`).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nSwitchStatus**: Returns additional information on the current status of the DALI control gear (see switchStatus [192]).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI208QueryUpSwitchOffThreshold**

This function block reads the variable `upSwitchOffThreshold [191]` from the DALI control gear.
The output of the DALI control gear is reset as soon as the rising virtual output value reaches or exceeds the threshold value `upSwitchOffThreshold`.

The threshold value is disabled if a value of 255 (MASK) is returned.

Prior to the DALI command QUERY UP SWITCH-OFF, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType [86]).

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
<td>Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nSwitchOffThreshold</td>
<td>BYTE</td>
<td>Threshold value (1...255) for switching off with a rising virtual output value.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI208QueryUpSwitchOnThreshold**

```plaintext
bStart  BOOL  bError
nAddress BYTE  ipResultMessage
eAddressType E_DALIAddressType  bBusy
eCommandPriority E_DALICommandPriority  nSwitchOffThreshold
```
This function block reads the variable `upSwitchOnThreshold` from the DALI control gear.

The output of the DALI control gear is reset as soon as the rising virtual output value reaches or exceeds the threshold value `upSwitchOnThreshold`.

The threshold value is disabled if a value of 255 (MASK) is returned.

Prior to the DALI command QUERY UP SWITCH-ON, the function block sends the DALI command ENABLE DEVICE TYPE 7, as is necessary for application extended commands (see also FB_DALI102EnableDeviceType).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).

- `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nSwitchOnThreshold</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nSwitchOnThreshold**: Threshold value (1…255) for switching on with a rising virtual output value.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>
### 3.1.2.4.4 Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables [191]</td>
<td>1</td>
<td>1…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [191]</td>
<td>255</td>
<td>1…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [191]</td>
<td>255</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [191]</td>
<td>0</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [191]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [192]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [192]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

?: undetermined

**upSwitchOnLevel**

This variable is the threshold value (1…255) for switching on the output with a rising virtual output value. The threshold value is disabled with the value 255 (MASK).

If bit 3 of features is set, the variable `upSwitchOnLevel` can be set with the function block `FB_DALI208SetUpSwitchOnThreshold [182]`.

The variable can be read with the function block `FB_DALI208QueryUpSwitchOnThreshold [189]`.

**upSwitchOffLevel**

This variable is the threshold value (1…255) for switching off the output with a rising virtual output value. The threshold value is disabled with the value 255 (MASK).

If bit 3 of features is set, the variable `upSwitchOffLevel` can be set with the function block `FB_DALI208SetUpSwitchOffThreshold [181]`.

The variable can be read with the function block `FB_DALI208QueryUpSwitchOffThreshold [188]`.

**downSwitchOnLevel**

This variable is the threshold value (0…255) for switching on the output with a falling virtual output value. The threshold value is disabled with the value 255 (MASK).

If bit 3 of features is set, the variable `downSwitchOnLevel` can be set with the function block `FB_DALI208SetDownSwitchOnThreshold [180]`.

The variable can be read with the function block `FB_DALI208QueryDownSwitchOnThreshold [184]`.

**downSwitchOffLevel**

This variable is the threshold value (0…255) for switching off the output with a falling virtual output value. The threshold value is disabled with the value 255 (MASK).

If bit 3 of features is set, the variable `downSwitchOffLevel` can be set with the function block `FB_DALI208SetDownSwitchOffThreshold [178]`.

The variable can be read with the function block `FB_DALI208QueryDownSwitchOffThreshold [183]`.

**features**

The variable contains information about the implemented properties of the DALI control gear.

The variable can be read with the function block `FB_DALI208QueryFeatures [185]`. 
### Part 209 (color/color temperature control)

**3.1.2.5.1 Introduction**

IEC 62386-209 describes DALI control gears for color or color temperature control. Currently, the Tc3_DALI library contains only function blocks for color temperature setting $T_C$ (Tunable-White), therefore the further explanations refer to this aspect.
**Representation of the color temperature**

The black-body line (BBL) or Planck curve runs within the CIE standard color chart, which represents the possible color space. Values on this curve correspond to a white hue specified in Kelvin. The higher the value, the colder the light is perceived.

Light sources from this range combine LEDs of different color temperatures on one board. This allows the lighting effect of, for example, Human Centric Lighting (HCL) to be adjusted throughout the day. It may be necessary to use different color temperatures for the presentation of different goods (e.g. food, colorful or bright backgrounds).

**Mirek and Kelvin**

In IEC 62386, the Mirek unit is used to specify the color temperature $T_C$. The function blocks of the Tc3_DALI library also use this unit. The color temperature values can be converted using the functions `KELVIN_TO_MIREK` and `MIREK_TO_KELVIN`.
3.1.2.5.2 Configuration commands

FB_DALI209SetPowerOnValues

The function block stores the values of the variables \(\text{nPowerOnLevel} \) [98] and \(\text{nColourTemperatureTc} \) [224] in the DALI control gear.

Initially, only the color temperature is adjustable.

- Prior to the DALI command SET POWER ON LEVEL, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application extended commands (see FB_DALI102EnableDeviceType [86]).

- The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

VAR_INPUT

- \(\text{bStart} \) : BOOL;
- \(\text{nAddress} \) : BYTE;
- \(\text{eAddressType} := \text{E_DALIAddressType.Short} \);
- \(\text{eCommandPriority} := \text{E_DALICommandPriority.MiddleLow} \);
- \(\text{nPowerOnLevel} \) : BYTE;
- \(\text{eColourType} := \text{E_DALIColourType.MASK} \);
- \(\text{nXCoordinate} \) : UINT;
- \(\text{nYCoordinate} \) : UINT;
- \(\text{nColourTemperatureTc} := 65534 \);
- \(\text{nPrimaryColour0} \) : UINT;
- \(\text{nPrimaryColour1} \) : UINT;
- \(\text{nPrimaryColour2} \) : UINT;
- \(\text{nPrimaryColour3} \) : UINT;
- \(\text{nPrimaryColour4} \) : UINT;
- \(\text{nPrimaryColour5} \) : UINT;
- \(\text{nRedDimlevel} \) : BYTE;
- \(\text{nGreenDimlevel} \) : BYTE;
- \(\text{nBlueDimlevel} \) : BYTE;
- \(\text{nWhiteDimlevel} \) : BYTE;
- \(\text{nAmberDimlevel} \) : BYTE;
- \(\text{nFreeColourDimlevel} \) : BYTE;
nWhiteDimlevel : BYTE;
nAmberDimlevel : BYTE;
nFreecolourDimlevel : BYTE;

\textbf{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress}: Address of a DALI device or a DALI group.

\textbf{eAddressType}: Defines whether the input \textit{nAddress} contains a short address (0…63) or a group address (0…15). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \textit{E_DALIAddressType} [547]).

\textbf{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \textit{E_DALICommandPriority} [549]).

\textit{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

\textbf{nPowerOnLevel}: Switch-on value (0…255).

\textbf{eColourType}: Defines the color setting with which the function block is operated (see \textit{Variables} [225]). MASK prevents the color representation from being changed. Currently, the function block only supports the setting of color temperature \textit{Tc} and MASK.

\textbf{nXCoordinate}: Entry of the \textit{x}-value of the color representation according to \textit{xy} color type (see \textit{X-coordinate} [224]) in the color setting \textit{Xy} coordinates (see \textit{Variables} [224]). Not currently implemented.

\textbf{nYCoordinate}: Entry of the \textit{y}-value of the color representation according to \textit{xy} color type (see \textit{Y-coordinate} [224]) in the color setting \textit{Xy} coordinates (see \textit{Variables} [224]). Not currently implemented.

\textbf{nColourTemperatureTc}: Entry of the desired color temperature (see \textit{Variables} [224]) in the setting \textit{Color temperature Tc} (see \textit{Variables} [224]).

\textbf{nPrimaryColourN}: Entry of the color representation on the corresponding output channel on the operating device in the color setting \textit{Primary color N} (see \textit{Variables} [224]). Not currently implemented.

\textbf{nRedDimlevel}: Setting of the red dimming level (see \textit{DimLevel} [224]) in the color setting RGBWAF (see \textit{Variables} [224]). Not currently implemented.

\textbf{nGreenDimlevel}: Setting of the green dimming level (see \textit{DimLevel} [224]) in the color setting RGBWAF (see \textit{Variables} [224]). Not currently implemented.

\textbf{nBlueDimlevel}: Setting of the blue dimming level (see \textit{DimLevel} [224]) in the color setting RGBWAF (see \textit{Variables} [224]). Not currently implemented.

\textbf{nWhiteDimlevel}: Setting of the white dimming level (see \textit{DimLevel} [224]) in the color setting RGBWAF (see \textit{Variables} [224]). Not currently implemented.

\textbf{nAmberDimlevel}: Setting of the amber dimming level (see \textit{DimLevel} [224]) in the color setting RGBWAF (see \textit{Variables} [224]). Not currently implemented.

\textbf{nFreecolourDimlevel}: Setting of the dimming level of the freely selectable color (see \textit{DimLevel} [224]) in the color setting RGBWAF (see \textit{Variables} [224]). Not currently implemented.

\textbf{VAR_OUTPUT}

\begin{verbatim}
  bError : BOOL;
  ipResultMessage : I_TcMessage;
  bBusy : BOOL;
\end{verbatim}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.
ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

FB_DALI209SetScene

The function block can be used to assign a specific lighting value to a scene.

Initially, only the color temperature is adjustable.

Prior to the DALI command SET SCENE, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see FB_DALI102EnableDeviceType [86]) and SET TEMPORARY COLOUR TEMPERATURE TC.

The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

VAR_INPUT

- bStart : BOOL;
- nAddress : BYTE;
- eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
- eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
- nSceneLevel : BYTE;
- nScene : BYTE;
- eColourType : E_DALIColourType := E_DALIColourType.MASK;
- nPrimaryColour0 : UINT;
- nPrimaryColour1 : UINT;
- nPrimaryColour2 : UINT;
- nPrimaryColour3 : UINT;
- nPrimaryColour4 : UINT;
- nPrimaryColour5 : UINT;
- nRedDimlevel : BYTE;
- nGreenDimlevel : BYTE;
- nBlueDimlevel : BYTE;
- nWhiteDimlevel : BYTE;
- nAmberDimlevel : BYTE;
- nFreckColourDimlevel : BYTE;
nXCoordinate : UINT;
nYCoordinate : UINT;
nColourTemperatureTc : UINT;
nPrimaryColour0 : UINT;
nPrimaryColour1 : UINT;
nPrimaryColour2 : UINT;
nPrimaryColour3 : UINT;
nPrimaryColour4 : UINT;
nPrimaryColour5 : UINT;
nRedDimlevel : BYTE;
nGreenDimlevel : BYTE;
nBlueDimlevel : BYTE;
nWhiteDimlevel : BYTE;
nAmberDimlevel : BYTE;
nFreecolourDimlevel : BYTE;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nSceneLevel: Output value of the desired scene (0…255).

nScene: Scene whose output value is to be changed (0…15).

eColourType: Defines the color setting with which the function block is operated (see Variables [225]). MASK prevents the color representation from being changed. Currently, the function block only supports the setting of color temperature Tc and MASK.

nXCoordinate: Entry of the x-value of the color representation according to xy color type (see X-coordinate [224]) in the color setting Xy coordinates (see Variables [224]). Not currently implemented.

nYCoordinate: Entry of the y-value of the color representation according to xy color type (see Y-coordinate [224]) in the color setting Xy coordinates (see Variables [224]). Not currently implemented.

nColourTemperatureTc: Entry of the desired color temperature (see Variables [224]) in the setting Color temperature Tc (see Variables [224]).

nPrimaryColourN: Entry of the color representation on the corresponding output channel on the operating device in the color setting Primary color N (see Variables [224]). Not currently implemented.

nRedDimlevel: Setting of the red dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nGreenDimlevel: Setting of the green dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nBlueDimlevel: Setting of the blue dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nWhiteDimlevel: Setting of the white dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nAmberDimlevel: Setting of the amber dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.
nFreecolourDimlevel: Setting of the dimming level of the freely selectable color (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI209SetSystemFailureValues**

The function block writes the color representation and light intensity (nSystemFailureLevel) of the light source to be used in the case of a system failure to the DALI control gear.

Initially, only the color temperature is adjustable.

Prior to the DALI command SET SYSTEM FAILURE LEVEL, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see FB_DALI102EnableDeviceType [86]) and SET TEMPORARY COLOUR TEMPERATURE TC.
The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

**VAR_INPUT**

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nSystemFailureLevel : BYTE;
eColourType : E.DALIColourType := E.DALIColourType.MASK;
xCoordinate : UINT;
yCoordinate : UINT;
colourTemperatureTc : UINT;
nPrimaryColour0 : UINT;
nPrimaryColour1 : UINT;
nPrimaryColour2 : UINT;
nPrimaryColour3 : UINT;
nPrimaryColour4 : UINT;
nPrimaryColour5 : UINT;
nRedDimLevel : BYTE;
nGreenDimLevel : BYTE;
nBlueDimLevel : BYTE;
nWhiteDimLevel : BYTE;
nAmberDimLevel : BYTE;
nFreeColourDimLevel : BYTE;

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).[547].

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).[549].

- **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**nSystemFailureLevel:** Output value in the case of a system failure (0…255).

**eColourType:** Defines the color setting with which the function block is operated (see Variables].[225]). MASK prevents the color representation from being changed. Currently, the function block only supports the setting of color temperature Tc and MASK.

**nXCoordinate:** Entry of the x-value of the color representation according to xy color type (see X-coordinate).[224]) in the color setting Xy coordinates (see Variables).[224]). Not currently implemented.

**nYCoordinate:** Entry of the y-value of the color representation according to xy color type (see Y-coordinate).[224]) in the color setting Xy coordinates (see Variables).[224]). Not currently implemented.

**nColourTemperatureTc:** Entry of the desired color temperature (see Variables).[224]) in the setting Color temperature Tc (see Variables).[224]).

**nPrimaryColourN:** Entry of the color representation on the corresponding output channel on the operating device in the color setting Primary color N (see Variables).[224]). Not currently implemented.

**nRedDimlevel:** Setting of the red dimming level (see DimLevel).[224]) in the color setting RGBWAF (see Variables).[224]). Not currently implemented.

**nGreenDimlevel:** Setting of the green dimming level (see DimLevel).[224]) in the color setting RGBWAF (see Variables).[224]). Not currently implemented.
nBlueDimlevel: Setting of the blue dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nWhiteDimlevel: Setting of the white dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nAmberDimlevel: Setting of the amber dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nFreecolourDimlevel: Setting of the dimming level of the freely selectable color (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

VAR_OUTPUT

| bError | : BOOL; |
| ipResultMessage | : I_TcMessage; |
| bBusy | : BOOL; |

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

FB_DALI209StoreColourTemperatureTcLimit

The function block saves the values of the variables colourTemperatureTcCoolest, colourTemperatureTcWarmest, colourTemperatureTcPhysicalCoolest or colourTemperatureTcPhysicalWarmest in the DALI control gear. The values can be read with function block FB_DALI209QueryColourValue [211].

- Prior to the DALI command STORE COLOUR TEMPERATURE Tc LIMIT, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also FB_DALI102EnableDeviceType [86]).

- The function block changes the DTR0, DTR1 and DTR2 of all connected DALI control gears.

- The function block sends the DALI command STORE COLOUR TEMPERATURE Tc LIMIT twice, as foreseen by the DALI standard for certain commands.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
eSelectLimitValue : E_DALIColourTemperatureTcLimit;
nValue : UINT;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

  eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

  


eSelectLimitValue: Specifies the limit value that is to be set (see E_DALIColourTemperatureTcLimit [548]).

nValue: The value that is written into the selected variable.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3.DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB_DALI209StoreGearFeaturesStatus

The function block saves the value in the variable gearFeatures/Status [225] of the DALI control gear. The value can be read with the function block FB_DALI209QueryGearFeaturesStatus [212].

If Bit 0 is set to 1, all commands for controlling the lamp power – with the exception of ENABLE DAPC SEQUENCE – automatically initiate a color transition.
Prior to the DALI command STORE GEAR FEATURES/STATUS, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also \texttt{FB\_DALI102EnableDeviceType} [86]).

The function block changes the DTR0 of all connected DALI control gears.

The function block sends the DALI command STORE GEAR FEATURES/STATUS twice, as foreseen by the DALI standard for certain commands.

### VAR\_INPUT

- \texttt{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.
- \texttt{nAddress}: Address of a DALI device or a DALI group.
- \texttt{eAddressType}: Defines whether the input \texttt{nAddress} contains a short address (0...63) or a group address (0...15). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E\_DALIAddressType} [547]).
- \texttt{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E\_DALICommandPriority} [549]). \texttt{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
- \texttt{nGearFeaturesStatus}: Value written to the variable \texttt{gearFeatures/Status} [225].

### VAR\_OUTPUT

- \texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.
- \texttt{ipResultMessage}: Interface pointer (see \texttt{Error evaluation} [574]) that can be used to obtain detailed information about the processing of the function block (see \texttt{Runtime messages} [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.
- \texttt{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.5.3 Control commands

FB_DALI209Activate

The command causes the buffered values of the temporary registers to be transferred to the lamp. It ends a running cross-fade and starts a new cross-fade for the respective color/color temperature.

Prior to the DALI command ACTIVATE, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also FB_DALI102EnableDeviceType [86]).

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.0</td>
<td>Tc3_DALI from v3.1.5.0</td>
</tr>
</tbody>
</table>
**FB_DALI209ColourTemperatureTcStepCooler**

Due to the function block the value *colourTemperatureTc* is reduced by 1 Mirek without cross-fading. If the *colourTemperatureTc* value already has the same value as *colourTemperatureTcCoolest*, no change takes place.

Bit 1 (Colour temperature Tc out of range) is set in *colourStatus* if the color temperature cannot be reached by the DALI control gear. This command is executed by the DALI control gear only if bit 5 (Colour type color temperature Tc active) is set in the *colourStatus* variable.

The functions *[KELVIN_TO_MIREK](#)* and *[MIREK_TO_KELVIN](#)* are available for converting from or to Kelvin.

Prior to the DALI command COLOUR TEMPERATURE Tc STEP COOLER, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also *[FB_DALI102EnableDeviceType](#)*).

### VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see *[E_DALIAddressType](#)*).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see *[E_DALICommandPriority](#)*).

- **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

- **ipResultMessage**: Interface pointer (see *[Error evaluation](#)* that can be used to obtain detailed information about the processing of the function block (see *[Runtime messages](#)*). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI209ColourTemperatureTcStepWarmer**

The `colourTemperatureTc` value is incremented by 1 Mirek without cross-fading. If the `colourTemperatureTc` value already has the same value as `colourTemperatureTcWarmest`, no change takes place.

Bit 1 (Color temperature Tc out of range) is set in `colourStatus` if the color temperature cannot be reached by the DALI control gear. This command is executed by the DALI control gear only if bit 5 (Color type color temperature Tc active) is set in the `colourStatus` variable.

The functions `KELVIN_TO_MIREK` and `MIREK_TO_KELVIN` are available for converting from or to Kelvin.

Prior to the DALI command COLOUR TEMPERATURE Tc STEP WARMER, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also `FB_DALI102EnableDeviceType`).

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI209CopyReportToTemporary**

The function block copies the contents of the variables with the color settings to the variables for the temporary color settings.

Prior to the DALI command COPY REPORT TO TEMPORARY, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

FB.DALI209SetColourTemperatureTc

Saves the value ColourTemperatureTc of the DALI control gear and, by executing the ACTIVATE command, causes the set color temperature to be assumed by the lamp.

The value can be read with the function block FB.DALI209QueryColourValue [211].

The value is expressed in units of 1 Mirek. A value of 0 is ignored and therefore not saved. The color temperature Tc can vary from 1 Mirek (1000000 K) to 65534 Mirek (15.26 K).

The functions KELVIN_TO_MIREK [220] and MIREK_TO_KELVIN [221] are available for converting from or to Kelvin.

Prior to the DALI command SET TEMPORARY COLOUR TEMPERATURE Tc, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also FB.DALI102EnableDeviceType [86]). To activate the color temperature, the function block sends the command ACTIVATE (see FB.DALI209Activate [203]).

The function block changes the DTR0 and the DTR1 of all connected DALI control gears.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E.DALIAddressType := E.DALIAddressType.Short;
eCommandPriority : E.DALICommandPriority := E.DALICommandPriority.MiddleLow;
nColourTemperatureTc : UINT;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E.DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E.DALICommandPriority [549]).

   eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nColourTemperatureTc: The value that is written to the variable ColourTemperatureTc and directly transferred to the output value (0…65534 Mirek, 65535).
**VAR_OUTPUT**

```plaintext
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
```

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI209SetTemporaryColourTemperatureTc**

```plaintext
VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nTemporaryColourTemperatureTc : UINT;
```

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**nTemporaryColourTemperatureTc**: Saves the value in the `temporaryColourTemperatureTc` variable of the DALI control gear. The value can be read with the function block `FB_DALI209QueryColourValue [211]`. The value is expressed in units of 1 Mirek. A value of 0 is ignored and therefore not saved. The color temperature Tc can vary from 1 Mirek (1000000 K) to 65534 Mirek (15.26 K).

The functions `KELVIN_TO_MIREK [220]` and `MIREK_TO_KELVIN [221]` are available for converting from or to Kelvin.

Prior to the DALI command SET TEMPORARY COLOUR TEMPERATURE Tc, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also `FB_DALI102EnableDeviceType [86]`).

The function block changes the DTR0 and the DTR1 of all connected DALI control gears.
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority[549]}).

\textit{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nTemporaryColourTemperatureTc: The value that is written to the variable \textit{temporaryColourTemperatureTc} (0…65534 Mirek, 65535).

\begin{tabular}{|l|}
\hline
\textbf{VAR_OUTPUT} \\
\hline
bError & \text{: BOOL;} \\
ipResultMessage & \text{: I_TcMessage;} \\
bBusy & \text{: BOOL;} \\
\hline
\end{tabular}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.

\textbf{ipResultMessage}: Interface pointer (see \textit{Error evaluation[574]}) that can be used to obtain detailed information about the processing of the function block (see \textit{Runtime messages[561]}). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.

\textbf{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

\begin{tabular}{|l|l|}
\hline
\textbf{Requirements} & \\
\hline
\textbf{Development environment} & \textbf{Required PLC library} \\
TwinCAT from v3.1.4024.10 & Tc3_DALI from v3.4.0.0 \\
\hline
\end{tabular}

3.1.2.5.4 Query commands

\textbf{FB_DALI209QueryColourStatus}

The function block reads the variable \textit{colourStatus[225]} from the DALI control gear.

\begin{itemize}
\item Prior to the DALI command QUERY COLOUR STATUS, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also \textit{FB_DALI102EnableDeviceType[86]}).
\end{itemize}

\begin{tabular}{|l|}
\hline
\textbf{VAR_INPUT} \\
\hline
bStart & \text{: BOOL;} \\
nAddress & \text{: BYTE;} \\
eAddressType & \text{E_DALIAAddressType := E_DALIAAddressType.Short;} \\
eCommandPriority & \text{E_DALICommandPriority := E_DALICommandPriority.MiddleLow;} \\
\hline
\end{tabular}

\textbf{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress}: Address of a DALI control gear or a DALI group.
**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

- **nColourStatus**: Output of the status information (see colourStatus [225]).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI209QueryColourTypeFeatures**

The function block reads the variable `colourTypeFeatures` [226] from the DALI control gear. This contains the color representations supported by the DALI control gear.

Prior to the DALI command QUERY COLOUR TYPE FEATURES, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also FB_DALI102EnableDeviceType [86]).

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

- **nAddress**: Address of a DALI control gear or a DALI group.
**eAddressType:** Defines whether the input \textit{nAddress} contains a short address (0…63) or a group address (0…15). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAddressType} [547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority} [549]).

\textit{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

\begin{quote}
\begin{tabular}{ll}
\texttt{bError} & : \texttt{BOOL}; \\
\texttt{ipResultMessage} & : \texttt{I_TcMessage}; \\
\texttt{bBusy} & : \texttt{BOOL}; \\
\texttt{nColourTypeFeatures} & : \texttt{BYTE}; \\
\end{tabular}
\end{quote}

\texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\texttt{ipResultMessage}: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\texttt{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

\texttt{nColourTypeFeatures}: Information about the color representations supported by the DALI control gear (see Variables [226]).

**Requirements**

\begin{center}
\begin{tabular}{|l|l|}
\hline
Development environment & Required PLC library \\
\hline
TwinCAT from v3.1.4024.10 & Tc3_DALI from v3.4.0.0 \\
\hline
\end{tabular}
\end{center}

**FB_DALI209QueryColourValue**

\begin{center}
\begin{tabular}{llllllll}
\texttt{bStart} & \texttt{BOOL} & \\
\texttt{nAddress} & \texttt{BYTE} & \\
\texttt{eAddressType} & \texttt{E_DALIAddressType} & \\
\texttt{eCommandPriority} & \texttt{E_DALICommandPriority} & \\
\texttt{eColourValue} & \texttt{E_DALIColourValue} & \\
\hline
\texttt{bError} & \texttt{BOOL} & \\
\texttt{ipResultMessage} & \texttt{I_TcMessage} & \\
\texttt{bBusy} & \texttt{BOOL} & \\
\texttt{nColourValue} & \texttt{UINT} & \\
\end{tabular}
\end{center}

The specified variable (color value) is read from the DALI control gear. The value to be read is defined by \texttt{eColourValue}.

Certain variables can be read directly by DALI commands (e.g. \texttt{FB_209QueryColourStatus} [209]). Further details on the variables can be found in section Variables [221].

The response must be 255 (MASK) if the control gear does not know the coordinates or if the primary color is not present.

\textit{i} Prior to the DALI command QUERY COLOUR VALUE, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see also \texttt{FB_DALI102EnableDeviceType} [86]).
The function block changes the DTR0 of all connected DALI control gears.

### VAR_INPUT

- **bStart**: BOOL; Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: BYTE; Address of a DALI device or a DALI group.
- **eAddressType**: E DALIAddressType := E DALIAddressType.Short; Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E DALIAddressType [547]).
- **eCommandPriority**: E DALICommandPriority := E DALICommandPriority.MiddleLow; Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E DALICommandPriority [549]).
  - eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
- **eColourValue**: E DALIColourValue; Parameter to be read from the DALI control gear (see E DALIColourValue [548]).

### VAR_OUTPUT

- **bError**: BOOL; This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: I_TcMessage; Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: BOOL; The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nColourValue**: UINT; Contains the value of the read parameter.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### FB.DALI209QueryGearFeaturesStatus

![FB.DALI209QueryGearFeaturesStatus](image)
The function block reads the variable `gearFeatures/Status` from the DALI control gear.

Prior to the DALI command QUERY GEAR FEATURES/STATUS, the function block sends the DALI command ENABLE DEVICE TYPE 8, as is necessary for application extended commands (see also `FB_DALI102EnableDeviceType`).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).
  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nGearFeatureStatus</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nGearFeatureStatus**: Output of the status information (see `gearFeatures/Status`).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
FB_DALI209QueryPowerOnValues

The function block reads the variables \texttt{powerOnLevel} \textsuperscript{98} and the supported values of the color settings from the DALI control gear.

Initially, only the color temperature is readable.

- In addition to the DALI command QUERY POWER ON LEVEL, the function block sends the DALI commands QUERY COLOUR VALUE and ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see \texttt{FB_DALI102EnableDeviceType} \textsuperscript{86}).

- The function block changes the DTR0 of all connected DALI control gears.

\begin{verbatim}
VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
\end{verbatim}

\textbf{bStart:} Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress:} Address of a DALI control gear or a DALI group.

\textbf{eAddressType:} Defines whether the input \textit{nAddress} contains a short address (0…63) or a group address (0…15). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAddressType} \textsuperscript{547}).

\textbf{eCommandPriority:} Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority} \textsuperscript{549}).

\textit{eCommandPriority} has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nPowerOnLevel : BYTE;
eColourType : E_DALIColourTypE;
nXCoordinate : UINT;
nYCoordinate : UINT;
nColourTemperatureTc : UINT;
nPrimaryColour0 : UINT;
nPrimaryColour1 : UINT;
nPrimaryColour2 : UINT;
nPrimaryColour3 : UINT;
nPrimaryColour4 : UINT;
nPrimaryColour5 : UINT;
nRedDimlevel : BYTE;
nGreenDimlevel : BYTE;
nBlueDimlevel : BYTE;
nWhiteDimlevel : BYTE;
nAmberDimlevel : BYTE;
nFreecolourDimlevel : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nPowerOnLevel: Output of the switch-on value (see powerOnLevel [98]) from the DALI control gear (0…255).

eColourType: Specifies the color setting with which the DALI control gear is operated (see Variables [225]).

nXCoordinate: Output of the x-value of the color representation according to xy color type (see X-coordinate [224]) in the color setting Xy coordinates (see Variables [224]). Not currently implemented.

nYCoordinate: Output of the y-value of the color representation according to xy color type (see Y-coordinate [224]) in the color setting Xy coordinates (see Variables [224]). Not currently implemented.

nColourTemperatureTc: Output of the color temperature (see Variables [224]) in the setting Color temperature Tc (see Variables [224]).

nPrimaryColourN: Output of the color representation on the corresponding output channel on the operating device in the color setting Primary color N (see Variables [224]). Not currently implemented.

nRedDimlevel: Output of the red dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nGreenDimlevel: Output of the green dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nBlueDimlevel: Output of the blue dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nWhiteDimlevel: Output of the white dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nAmberDimlevel: Output of the amber dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.
nFreecolourDimLevel: Output of the dimming level of the freely selectable color (see DimLevel \[224\]) in the color setting RGBWAF (see Variables \[224\]). Not currently implemented.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

FB_DALI209QuerySceneValues

The function block reads the output values for the specified scene nScene from the DALI control gear. Initially, only the color temperature is readable.

In addition to the DALI command QUERY SCENE LEVEL, the function block sends the DALI commands QUERY COLOUR VALUE and ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see FB_DALI102EnableDeviceType \[86\]).

The function block changes the DTR0 of all connected DALI control gears.

VAR_INPUT

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType \[547\]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see [E_DALICommandPriority](#) [549]).

*Note:* `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**nScene**: Scene whose output value is to be read (0...15).

---

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nSceneLevel</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eColourType</td>
<td>E_DALIColourType;</td>
</tr>
<tr>
<td>nXCoordinate</td>
<td>UINT;</td>
</tr>
<tr>
<td>nYCoordinate</td>
<td>UINT;</td>
</tr>
<tr>
<td>nColourTemperatureTc</td>
<td>UINT;</td>
</tr>
<tr>
<td>nPrimaryColour0</td>
<td>UINT;</td>
</tr>
<tr>
<td>nPrimaryColour1</td>
<td>UINT;</td>
</tr>
<tr>
<td>nPrimaryColour2</td>
<td>UINT;</td>
</tr>
<tr>
<td>nPrimaryColour3</td>
<td>UINT;</td>
</tr>
<tr>
<td>nPrimaryColour4</td>
<td>UINT;</td>
</tr>
<tr>
<td>nPrimaryColour5</td>
<td>UINT;</td>
</tr>
<tr>
<td>nRedDimlevel</td>
<td>BYTE;</td>
</tr>
<tr>
<td>nGreenDimlevel</td>
<td>BYTE;</td>
</tr>
<tr>
<td>nBlueDimlevel</td>
<td>BYTE;</td>
</tr>
<tr>
<td>nWhiteDimlevel</td>
<td>BYTE;</td>
</tr>
<tr>
<td>nAmberDimlevel</td>
<td>BYTE;</td>
</tr>
<tr>
<td>nFreecolourDimlevel</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see [error evaluation](#) [574]) that can be used to obtain detailed information about the processing of the function block (see [runtime messages](#) [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nSceneLevel**: Output value of the desired scene (0...255).

**eColourType**: Specifies the color setting with which the DALI control gear is operated (see [Variables](#) [225]).

**nXCoordinate**: Output of the x-value of the color representation according to xy color type (see [X-coordinate](#) [224]) in the color setting Xy coordinates (see [Variables](#) [224]). Not currently implemented.

**nYCoordinate**: Output of the y-value of the color representation according to xy color type (see [Y-coordinate](#) [224]) in the color setting Xy coordinates (see [Variables](#) [224]). Not currently implemented.

**nColourTemperatureTc**: Output of the color temperature (see [Variables](#) [224]) in the setting Color temperature Tc (see [Variables](#) [224]).

**nPrimaryColourN**: Output of the color representation on the corresponding output channel on the operating device in the color setting Primary color N (see [Variables](#) [224]). Not currently implemented.

**nRedDimlevel**: Output of the red dimming level (see [DimLevel](#) [224]) in the color setting RGBWAF (see [Variables](#) [224]). Not currently implemented.

**nGreenDimlevel**: Output of the green dimming level (see [DimLevel](#) [224]) in the color setting RGBWAF (see [Variables](#) [224]). Not currently implemented.
**nBlueDimlevel**: Output of the blue dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

**nWhiteDimlevel**: Output of the white dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

**nAmberDimlevel**: Output of the amber dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

**nFreecolourDimlevel**: Output of the dimming level of the freely selectable color (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**FB_DALI209QuerySystemFailureValues**

The function block reads the variable `systemFailureLevel` and the corresponding color/color temperature settings from the DALI control gear.

Initially, only the color temperature is readable.

- In addition to the DALI command QUERY SYSTEM FAILURE LEVEL, the function block sends the DALI commands QUERY COLOUR VALUE and ENABLE DEVICE TYPE 8, as is necessary for application-related extension commands (see `FB_DALI102EnableDeviceType`).

- The function block changes the DTR0 of all connected DALI control gears.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nSystemFailureLevel : BYTE;
eColourType : E_DALIColourType;
nXCoordinate : UINT;
nYCoordinate : UINT;
nColourTemperatureTc : UINT;
nPrimaryColour0 : UINT;
nPrimaryColour1 : UINT;
nPrimaryColour2 : UINT;
nPrimaryColour3 : UINT;
nPrimaryColour4 : UINT;
nPrimaryColour5 : UINT;
nRedDimlevel : BYTE;
nGreenDimlevel : BYTE;
nBlueDimlevel : BYTE;
nWhiteDimlevel : BYTE;
nAmberDimlevel : BYTE;
nFreecolourDimlevel : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nSystemFailureLevel: Output value in the case of a system failure (0…255).

eColourType: Specifies the color setting with which the DALI control gear is operated (see Variables [225]).

nXCoordinate: Output of the x-value of the color representation according to xy color type (see X-coordinate [224]) in the color setting Xy coordinates (see Variables [224]). Not currently implemented.

nYCoordinate: Output of the y-value of the color representation according to xy color type (see Y-coordinate [224]) in the color setting Xy coordinates (see Variables [224]). Not currently implemented.

nColourTemperatureTc: Output of the color temperature (see Variables [224]) in the setting Color temperature Tc (see Variables [224]).
nPrimaryColourN: Output of the color representation on the corresponding output channel on the operating
device in the color setting Primary color N (see Variables [224]). Not currently implemented.

nRedDimlevel: Output of the red dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nGreenDimlevel: Output of the green dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nBlueDimlevel: Output of the blue dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nWhiteDimlevel: Output of the white dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nAmberDimlevel: Output of the amber dimming level (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

nFreecolourDimlevel: Output of the dimming level of the freely selectable color (see DimLevel [224]) in the color setting RGBWAF (see Variables [224]). Not currently implemented.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.2.5.5 Help Functions

KELVIN_TO_MIREK

Conversion of the color temperature from Kelvin to Mirek.

Mirek is a color temperature unit used by most DALI commands. The return value of the function is limited to the value range 0…65535 (see table).

Mirek = 1000000 / (color temperature in Kelvin).

<table>
<thead>
<tr>
<th>Kelvin</th>
<th>Mirek</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>65535</td>
</tr>
<tr>
<td>15.0</td>
<td>65535</td>
</tr>
<tr>
<td>16.0</td>
<td>62500</td>
</tr>
<tr>
<td>1000.0</td>
<td>1000</td>
</tr>
<tr>
<td>10000.0</td>
<td>100</td>
</tr>
<tr>
<td>1000000.0</td>
<td>1</td>
</tr>
<tr>
<td>10000001.0</td>
<td>0</td>
</tr>
</tbody>
</table>

VAR_INPUT
fKelvin : LREAL;
fKelvin: Color temperature in Kelvin.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

MIREK_TO_KELVIN

Conversion of the color temperature from Mirek to Kelvin.

Mirek is a color temperature unit used by most DALI commands. The return value of the function is limited to the value range 0...1000001 (see table).

Kelvin = 1000000 / (color temperature in Mirek).

<table>
<thead>
<tr>
<th>Mirek</th>
<th>Kelvin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000001.0</td>
</tr>
<tr>
<td>1</td>
<td>1000000.0</td>
</tr>
<tr>
<td>100</td>
<td>10000.0</td>
</tr>
<tr>
<td>1000</td>
<td>1000.0</td>
</tr>
<tr>
<td>10000</td>
<td>100.0</td>
</tr>
<tr>
<td>65534</td>
<td>15.259</td>
</tr>
<tr>
<td>65535</td>
<td>0.0</td>
</tr>
</tbody>
</table>

VAR_INPUT

nMirek : UINT;

nMirek: Color temperature in Mirek.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.5.6 Variables

Every DALI control gear for color/color temperature control has a certain number of variables (parameters) from which it is possible to read a variety of information or to modify individual parameters.

Some variables can be read directly by DALI commands (e.g. FB_DALI209QueryColourStatus [209]). The function block FB_DALI209QueryColourValue [211] can be used to read further variables.
<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Variables</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>temporaryX-Coordinate</td>
<td>65535</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0, 2</td>
<td></td>
</tr>
<tr>
<td>reportX-Coordinate</td>
<td>65535</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>No change</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>temporaryY-Coordinate</td>
<td>65535</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0, 2</td>
<td></td>
</tr>
<tr>
<td>reportY-Coordinate</td>
<td>65535</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>No change</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>temporaryColourTemperatureTc</td>
<td>65535</td>
<td>1…65535</td>
<td>2 bytes</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>No change</td>
<td>1…65535</td>
<td>2 bytes</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>colourTemperatureTcCoolest</td>
<td>colourTemperature TcPhysicalCoolest colourTemperature TcPhysicalCoolest colourTemperature TcPhysicalWarmest, 65535</td>
<td>2 bytes</td>
<td>1</td>
<td>Read only</td>
<td></td>
</tr>
<tr>
<td>colourTemperatureTcWarmest</td>
<td>colourTemperature TcPhysicalWarmest colourTemperature TcPhysicalWarmest colourTemperature TcPhysicalWarmest, 65535</td>
<td>2 bytes</td>
<td>1</td>
<td>Read only</td>
<td></td>
</tr>
<tr>
<td>colourTemperatureTcPhysicalCoolest</td>
<td>No change</td>
<td>colourTemperature TcPhysicalCoolest colourTemperature TcPhysicalWarmest, 65535</td>
<td>2 bytes</td>
<td>1</td>
<td>Read only</td>
</tr>
<tr>
<td>colourTemperatureTcPhysicalWarmest</td>
<td>No change</td>
<td>colourTemperature TcPhysicalCoolest colourTemperature TcPhysicalWarmest, 65535</td>
<td>2 bytes</td>
<td>1</td>
<td>Read only</td>
</tr>
<tr>
<td>temporaryPrimaryNDimlevel</td>
<td>65535</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>reportPrimaryNDimlevel</td>
<td>65535</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>primaryNDimlevel</td>
<td>No change</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>x-CoordinatePrimaryN</td>
<td>No change</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0, 2</td>
<td>Read only</td>
</tr>
<tr>
<td>y-CoordinatePrimaryN</td>
<td>No change</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0, 2</td>
<td>Read only</td>
</tr>
<tr>
<td>tyPrimary</td>
<td>No change</td>
<td>0…65535</td>
<td>2 bytes</td>
<td>0, 2</td>
<td>Read only</td>
</tr>
<tr>
<td>temporaryRedDimlevel</td>
<td>255</td>
<td>0…255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>reportRedDimlevel</td>
<td>255</td>
<td>0…255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>temporaryGreenDimlevel</td>
<td>255</td>
<td>0…255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Reset value</td>
<td>Scope</td>
<td>Size</td>
<td>Variables [224]</td>
<td>Comment</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>------------</td>
<td>--------</td>
<td>-----------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>reportGreenDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>temporaryBlueDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>reportBlueDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>temporaryWhiteDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>reportWhiteDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>temporaryAmberDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>reportAmberDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>temporaryFreecolourDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>reportFreecolourDimlevel</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>temporaryRGBWF Control</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>report RGBWF Control</td>
<td>255</td>
<td>0...255</td>
<td>1 byte</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>scene0-15ColourValue</td>
<td>65535</td>
<td>0...65535</td>
<td>32 bytes...192 bytes</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [225]</td>
<td>255</td>
<td>16#10, 16#20, 16#40, 16#80, 16#FF</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Variables [225]</td>
<td>255</td>
<td>16#10, 16#20, 16#40, 16#80, 16#FF</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Variables [225]</td>
<td>65535</td>
<td>16#10, 16#20, 16#40, 16#80, 16#FF</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [225]</td>
<td>Manufacturer-dependent</td>
<td>16#10, 16#20, 16#40, 16#80, 16#FF</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [225]</td>
<td>Manufacturer-dependent</td>
<td>0...65535</td>
<td>2 bytes...12 bytes</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [225]</td>
<td>Manufacturer-dependent</td>
<td>16#10, 16#20, 16#40, 16#80, 16#FF</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [225]</td>
<td>Manufacturer-dependent</td>
<td>0...65535</td>
<td>2 bytes...12 bytes</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
<tr>
<td>Name</td>
<td>Reset value</td>
<td>Scope</td>
<td>Size</td>
<td>Variables</td>
<td>Comment</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>------------------------</td>
<td>--------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>Variables [225]</td>
<td>2#XX00_0001</td>
<td>2#XX00_0000, 2#XX00_0001</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Variables [225]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Variables [226]</td>
<td>No change</td>
<td>0…255</td>
<td>1 byte</td>
<td>0, 1, 2, 3</td>
<td>Read only</td>
</tr>
</tbody>
</table>

? or X: not specified

In the case of 1-byte values the value 255 is also called MASK.

In the case of 2-byte values the value 65535 is also called MASK.

**Necessary color representation**

Specifies the color representation that the DALI control gear must support so that it contains the appropriate variables:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xy coordinates</td>
</tr>
<tr>
<td>1</td>
<td>Color temperature Tc</td>
</tr>
<tr>
<td>2</td>
<td>Primary (color) N</td>
</tr>
<tr>
<td>3</td>
<td>RGBWAF</td>
</tr>
</tbody>
</table>

**x-Coordinate**

Chromaticity coordinate x in the xy color type

The x-Coordinate is specified in MIREK_TO_KELVIN [221].

It can vary from 0 Mirek (0 K) to 65534 Mirek (15.26 K).

At a value of 65535 (MASK), the value is not defined.

**y-Coordinate**

Chromaticity coordinate y in the xy color type

The y-Coordinate is specified in MIREK_TO_KELVIN [221].

It can vary from 0 Mirek (0 K) to 65534 Mirek (15.26 K).

At a value of 65535 (MASK), the value is not defined.

**colourTemperatureTc**

The variable colourTemperatureTc defines the color temperature. It is adjustable in the range between colourTemperatureTcWarmest and colourTemperatureTcCoolest.

The colourTemperatureTc is specified in MIREK_TO_KELVIN [221].

It can vary from 1 Mirek (1,000,000 K) to 65534 Mirek (15.265 K).

**dimlevel**

Setting of the dimming level of the corresponding color (red, green, blue, white, amber or a freely selectable color) from 0 to 255.

**assignedColour**

The association between output channel and color is defined in the assignedColour variable. Each byte contains the color of the corresponding channel. The function block FB_DALI209QueryAssignedColour can be used to read out the value.
**colourType**

The `colourType` variable defines the color representations supported by the DALI control gear. The function block `FB_DALI209QueryColourValue` can be used to read out the values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No color assigned</td>
</tr>
<tr>
<td>1</td>
<td>Red</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
</tr>
<tr>
<td>3</td>
<td>Blue</td>
</tr>
<tr>
<td>4</td>
<td>White</td>
</tr>
<tr>
<td>5</td>
<td>Amber</td>
</tr>
<tr>
<td>6</td>
<td>Freely selectable color</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>E_DALIColourType</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XyCoordinate</td>
<td>xy coordinates</td>
</tr>
<tr>
<td>ColourTemperature</td>
<td>Color temperature Tc</td>
</tr>
<tr>
<td>PrimaryNDimLevel</td>
<td>Primary (color) N</td>
</tr>
<tr>
<td>RGBWAFControl</td>
<td>RGBWAF</td>
</tr>
<tr>
<td>MASK</td>
<td>No color change</td>
</tr>
</tbody>
</table>

**colourStatus**

`colourStatus` contains information about the current status of the DALI control gear.

The function block `FB_DALI209QueryColourStatus` can be used to read out the values.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xy coordinate color point is outside the valid range.</td>
</tr>
<tr>
<td>1</td>
<td>Color temperature Tc lies outside the valid range.</td>
</tr>
<tr>
<td>2</td>
<td>Automatic calibration is active.</td>
</tr>
<tr>
<td>3</td>
<td>Automatic calibration was successful.</td>
</tr>
<tr>
<td>4</td>
<td>Color representation xy-coordinate active.</td>
</tr>
<tr>
<td>5</td>
<td>Color representation color temperature Tc active.</td>
</tr>
<tr>
<td>6</td>
<td>Color representation primary N active.</td>
</tr>
<tr>
<td>7</td>
<td>Color representation RGBWAF active.</td>
</tr>
</tbody>
</table>

**gearFeatures/Status**

`gearFeatures/Status` contains information about the current status of the DALI control gear.

The function block `FB_DALI209QueryGearFeaturesStatus` can be used to read out the values. The function block `FB_DALI209StoreGearFeaturesStatus` can be used to write the values.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatic activation</td>
</tr>
<tr>
<td>1...5</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Automatic calibration is supported</td>
</tr>
<tr>
<td>7</td>
<td>Restoration of the automatic calibration is supported</td>
</tr>
</tbody>
</table>
**RGBWAFControl**

*RGBWAFControl* contains further information about the assignment between output channel and color.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output channel 0 / red</td>
</tr>
<tr>
<td>1</td>
<td>Output channel 1 / green</td>
</tr>
<tr>
<td>2</td>
<td>Output channel 2 / blue</td>
</tr>
<tr>
<td>3</td>
<td>Output channel 3 / white</td>
</tr>
<tr>
<td>4</td>
<td>Output channel 4 / amber</td>
</tr>
<tr>
<td>5</td>
<td>Output channel 5 / free selectable color</td>
</tr>
</tbody>
</table>
| 6…7| 00 = channel control  
     | 01 = color control          |
|     | 10 = standardized color control |
|     | 11 = reserved               |

**colourTypeFeatures**

colourTypeFeatures defines the color representations supported by the DALI control gear. The values can be read out using the `FB_DALI209QueryColourTypeFeatures` function block.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The DALI control gear supports color representation through xy coordinates.</td>
</tr>
<tr>
<td>1</td>
<td>The DALI control gear supports color representation through color temperature Tc.</td>
</tr>
<tr>
<td>2…4</td>
<td>Number of primary colors supported by the DALI control gear. A value of 0 means that this color representation by primary colors is not supported.</td>
</tr>
<tr>
<td>5…7</td>
<td>Number of RGBWAF channels supported by the DALI control gear. A value of 0 means that this color representation by RGBWAF is not supported.</td>
</tr>
</tbody>
</table>

### 3.1.2.6 Part 251 (memory bank 1 extension)

#### 3.1.2.6.1 Introduction

In specifications that go beyond the current DALI-2 standard, the Digital Illumination Interface Alliance (DiiA) describes additional DALI parts for the operation of intelligent luminaires.

These new parts are combined under the term D4i (DALI inside of luminaires).

The current library supports Parts 251 (DT50), 252 (DT51) and 253 (DT52), which describe the location, type and format of the data in the memory banks of the DALI control gears. In addition, they describe the exchange of data with DALI control devices.

**DALI Part 251 – memory bank 1 extension**

Part 251 extends previously unoccupied areas of memory bank 1 for vendor-specific information. This information does not change the functionality of the control gear.

The data stored here are directly available to the DALI network for technical management, e.g. in large properties.

Part 251 provides information about the inventory, such as device number, date of manufacture and color of the device. Information on rated power, rated illuminance, rated voltage, minimum power consumption, color rendering index and correlated color temperature are also available.

The values in Part 251 are read-only.

The entire contents of memory bank 1 are read via the function block `FB_DALI102ReadMemoryBank`.

Part 251 also provides the function blocks described below for reading the individual parameters.
3.1.2.6.2 FB_DALI251QueryColourRenderingIndex

The function block reads the color rendering index from the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- *eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nColourRenderingIndex</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nColourRenderingIndex**: Outputs the color rendering index (0…100).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the correlated color temperature from the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nCorrelatedColourTemperature**: Outputs the correlated color temperature [K] (0...17000). If the value 16#FFFE is returned, then Part 209 is not implemented in the DALI control gear.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
### FB_DALI251QueryLuminaireColour

The function block reads the luminaire color from the DALI control gear.

#### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input **nAddress** contains a short address (0…63) or a group address (0…15). Input **nAddress** has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

**eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

#### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **sLuminaireColour**: Outputs the luminaire color.

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.6.5 FB_DALI251QueryLuminaireDateOfManufacture

The function block reads the year and week of production from the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nYear</td>
<td>Year of production (2000…2099).</td>
</tr>
<tr>
<td>nWeek</td>
<td>Week of production (1…53).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development Environment</th>
<th>Required PLC Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
### FB_DALI251QueryLuminaireIdentification

The function block reads the identification designation of the luminaire from the DALI control gear.

#### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]). eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.</td>
</tr>
</tbody>
</table>

#### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipErrorMessage. The output is set to FALSE as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipErrorMessage</td>
<td>I_TcMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>sLuminaireIdentification</td>
<td>STRING(60)</td>
<td>Identification designation.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.6.7 FB_DALI251QueryNominalInputPower

The function block reads the nominal input power of the luminaire from the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nNominalInputPower : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nNominalInputPower: Nominal power [W] (0…65534).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.6.8 FB_DALI251QueryNominalLightOutput

The function block reads the nominal light output of the luminaire from the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nNominalLightOutput</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nNominalLightOutput**: Nominal illuminance [Lm] (0…16777214).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.6.9 FB_DALI251QueryNominalMaximumACMainsVoltage

The function block reads the maximum nominal mains input voltage of the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nNominalMaximumACMainsVoltage : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nNominalMaximumACMainsVoltage: Maximum nominal input voltage [V] (90…480).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the minimum nominal mains input voltage of the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nNominalMinimumACMainsVoltage**: Minimum nominal input voltage [V] (90…480).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.6.11  FB_DALI251QueryPowerAtMinimumDimLevel

The function block reads the power consumption at the minimum dimming level of the DALI control gear.

**VAR_INPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

**VAR_OUTPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nPowerAtMinimumDimLevel</td>
<td>UINT</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nPowerAtMinimumDimLevel:** Power consumption [W] at the minimum dimming level (0…65534).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.7 Part 252 (energy reporting)

3.1.2.7.1 Introduction

In specifications that go beyond the current DALI-2 standard, the Digital Illumination Interface Alliance (DiiA) describes additional DALI parts for the operation of intelligent luminaires.

These new parts are combined under the term D4i (DALI inside of luminaires).

The current library supports Parts 251 (DT50), 252 (DT51) and 253 (DT52).

These parts contain functions that can be forwarded to the higher-level DALI network for evaluation and control.

They describe the location, type and format of the data in the memory banks of the DALI control gears. In addition, they describe the exchange of data with DALI control devices.

Part 252 – Energy reporting

Part 252 describes additional memory banks in which energy and consumption data can be read.

Memory bank 202 contains information about active energy and active power. This information is obligatory and is provided by DALI control gears that support Part 252.

Memory bank 203 contains information on apparent energy and apparent power.

Memory bank 204 contains information on output energy and output power.

The information from memory banks 203 and 204 is optional and does not need to be provided by DALI control gears that support Part 252.

The values in Part 252 are read-only.

The entire contents of memory banks 202, 203 and 204 are read via the function block FB_DALI102ReadMemoryBank [31].

Part 252 also provides the function blocks described below for reading the individual parameters. The actual value of the read parameter is calculated from the read output value and the scaling factor. The function blocks output both values for further calculation.

3.1.2.7.2 FB_DALI252QueryActiveEnergy

The function block reads the active energy from the DALI control gear.

It is calculated as follows:

\[ E = n_{ActiveEnergy} \times 10^{n_{ScaleFactorForActiveEnergy}} \]

VAR_INPUT

| bStart | nAddress | eAddressType | eCommandPriority |
| ------ | -------- | ------------ | -----------------
| BOOL  | BYTE    | E_DALIAddressType | E_DALICommandPriority |

The entire contents of memory banks 202, 203 and 204 are read via the function block FB_DALI102ReadMemoryBank.
bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

  eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nActiveEnergy</td>
<td>ULINT</td>
</tr>
<tr>
<td>nScaleFactorForActiveEnergy</td>
<td>SINT</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nActiveEnergy: Active energy [Wh] (0…281474976710653).

nScaleFactorForActiveEnergy: Scaling factor for the active energy (-6...6).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.7.3  FB_DALI252QueryActiveEnergyLoadside

The function block reads the load-side active energy from the DALI control gear.

It is calculated as follows:

\[ E_{Loadside} = nActiveEnergyLoadside \times 10^{nScaleFactorForLoadsideEnergy} \]
VAR_INPUT

| bStart     : BOOL;  |
| nAddress   : BYTE;  |
| eAddressType : E_DALIAddressType := E_DALIAddressType.Short; |
| eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow; |

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

| bError                  : BOOL;  |
| ipResultMessage         : I_TcMessage;  |
| bBusy                   : BOOL;  |
| nActiveEnergyLoadside   : ULINT;  |
| nScaleFactorForLoadsideEnergy : SINT;  |

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nActiveEnergyLoadside: Load-side active energy [Wh] (0…281474976710653).

nScaleFactorForLoadsideEnergy: Scaling factor for the load-side active energy (-6...6).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.7.4 FB_DALI252QueryActivePower

The function block reads the active power from the DALI control gear.

It is calculated as follows:
\[
P = n\text{ActivePower} \times 10^{n\text{ScaleFactorForActivePower}}
\]

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input \(n\text{Address}\) contains a short address (0…63) or a group address (0…15). Input \(n\text{Address}\) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \(\text{E_DALIAddressType}\)).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \(\text{E_DALICommandPriority}\)).

\(e\text{CommandPriority}\) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nActivePower</td>
<td>UDINT</td>
</tr>
<tr>
<td>nScaleFactorForActivePower</td>
<td>SINT</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \(ip\text{ResultMessage}\). The output is set to FALSE again as soon as \(b\text{Busy}\) switches to TRUE.
- **ipResultMessage**: Interface pointer (see \(\text{error evaluation}\)) that can be used to obtain detailed information about the processing of the function block (see \(\text{runtime messages}\)). The interface pointer is valid after \(b\text{Busy}\) has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nActivePower**: Active power [W] (0…4294967293).
- **nScaleFactorForActivePower**: Scaling factor for the active power (-6…6).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**3.1.2.7.5 FB_DALI252QueryActivePowerLoadside**

The function block reads the load-side active power from the DALI control gear.
It is calculated as follows:

\[ P_{\text{Loadside}} = n\text{ActivePowerLoadside} \times 10^{n\text{ScaleFactorForLoadsidePower}} \]

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

`eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nActivePowerLoadside</td>
<td>UDINT</td>
</tr>
<tr>
<td>nScaleFactorForLoadsidePower</td>
<td>SINT</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nActivePowerLoadside**: Load-side active power [W] (0...4294967293).

**nScaleFactorForLoadsidePower**: Scaling factor for the load-side active power (-6...6).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

**3.1.2.7.6 FB_DALI252QueryApparentEnergy**
The function block reads the apparent energy from the DALI control gear.

It is calculated as follows:

\[
\text{ApparentEnergy} = n\text{ApparentEnergy} \times 10^{n\text{ScaleFactorForApparentEnergy}}
\]

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart : BOOL;</td>
</tr>
<tr>
<td>nAddress : BYTE;</td>
</tr>
<tr>
<td>eAddressType : E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input *nAddress* contains a short address (0...63) or a group address (0...15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see *E_DALIAddressType* [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see *E_DALICommandPriority* [549]).

*eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError : BOOL;</td>
</tr>
<tr>
<td>ipResultMessage : I_TcMessage;</td>
</tr>
<tr>
<td>bBusy : BOOL;</td>
</tr>
<tr>
<td>nApparentEnergy : ULINT;</td>
</tr>
<tr>
<td>nScaleFactorForApparentEnergy : SINT;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see *error evaluation* [574]) that can be used to obtain detailed information about the processing of the function block (see *runtime messages* [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nApparentEnergy**: Apparent energy [VAh] (0...281474976710653).

**nScaleFactorForApparentEnergy**: Scaling factor for the apparent energy (-6...6).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.7.7 FB_DALI252QueryApparentPower

The function block reads the apparent power from the DALI control gear.

It is calculated as follows:

\[
S = n_{\text{ApparentPower}} \times 10^{n_{\text{ScaleFactorForApparentPower}}}
\]

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input \( n_{\text{Address}} \) contains a short address (0…63) or a group address (0…15). Input \( n_{\text{Address}} \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( E_{\text{DALIAddressType}} \) [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( E_{\text{DALICommandPriority}} \) [549]).

\( e_{\text{CommandPriority}} \) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( ip_{\text{ResultMessage}} \). The output is set to FALSE again as soon as \( b_{\text{Busy}} \) switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \( b_{\text{Busy}} \) has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nApparentPower**: Apparent power [VA] (0…4294967293).
- **nScaleFactorForApparentPower**: Scaling factor for the apparent power (-6...6).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8 Part 253 (diagnostics and maintenance)

3.1.2.8.1 Introduction

In specifications that go beyond the current DALI-2 standard, the Digital Illumination Interface Alliance (DiiA) describes additional DALI parts for the operation of intelligent luminaires.

These new parts are combined under the term D4i (DALI inside of luminaires).

The current library supports Parts 251 (DT50), 252 (DT51) and 253 (DT52).

These parts contain functions that can be forwarded to the higher-level DALI network for evaluation and control.

They describe the location, type and format of the data in the memory banks of the DALI control gears. In addition, they describe the exchange of data with DALI control devices.

Part 253 – Diagnostics and maintenance

Part 252 writes additional memory banks in which diagnostic and maintenance data can be read and partly specified.

Memory bank 205 contains information on diagnostics and maintenance of the DALI control gear.

Memory bank 206 contains information on the diagnostics and maintenance of the light source.

Memory bank 207 contains maintenance information on the luminaire.

This information is obligatory and is provided by DALI control gears that support Part 253.

The entire contents of memory banks 205, 206 and 207 are read via the function block `FB_DALI102ReadMemoryBank`.

Part 253 also provides the function blocks described below for reading the individual parameters.

3.1.2.8.2 FB_DALI253QueryControlGearExternalSupplyOvervoltage

The function block indicates whether an overvoltage has been detected in the external supply voltage of the DALI control gear.

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E.DALICommandPriority := E.DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAAddressType`).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E.DALICommandPriority`).
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bControlGearExternalSupplyOvervoltage : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation[574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages[561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bControlGearExternalSupplyOvervoltage: The output is set as soon as an overvoltage has been detected in the external supply voltage of the DALI control gear.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.8.3 FB_DALI253QueryControlGearExternalSupplyOvervoltageCounter

The function block reads the counter for the detected overvoltage events of the external supply voltage of the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType[547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority[549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
**VAR_OUTPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nControlGearExternalSupplyOvervoltageCounter</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage:** Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nControlGearExternalSupplyOvervoltageCounter:** Number of detected overvoltage events of the external supply voltage (0...253).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.8.4 **FB_DALI253QueryControlGearExternalSupplyUndervoltage**

The function block indicates whether an undervoltage has been detected in the external supply voltage of the DALI control gear.

**VAR_INPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input *nAddress* contains a short address (0...63) or a group address (0...15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bControlGearExternalSupplyUndervoltage</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bControlGearExternalSupplyUndervoltage: The output is set as soon as an undervoltage has been detected in the external supply voltage of the DALI control gear.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.8.5  FB_DALI253QueryControlGearExternalSupplyUndervoltageCounter

The function block reads the counter for the detected undervoltage events of the external supply voltage of the DALI control gear.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nControlGearExternalSupplyUndervoltageCounter</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nControlGearExternalSupplyUndervoltageCounter: Number of detected undervoltage events of the external supply voltage (0...253).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.8.6 FB.DALI253QueryControlGearExternalSupplyVoltage

The function block reads the effective value of the external supply voltage from the DALI control gear.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>fControlGearExternalSupplyVoltage</td>
<td>LREAL</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**fControlGearExternalSupplyVoltage:** RMS value of the external supply voltage [Vrms] (0.0…6553.3).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.2.8.7  FB_DALI253QueryControlGearExternalSupplyVoltageFrequency

**VAR_INPUT**

- **bStart** : BOOL;
- **nAddress** : BYTE;
- **eAddressType** : E_DALIAddressType := E_DALIAddressType.Short;
- **eCommandPriority** : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input **nAddress** contains a short address (0…63) or a group address (0…15). Input **nAddress** has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see **E_DALIAddressType** [547]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see **E_DALICommandPriority** [549]).

**VAR_OUTPUT**

- **bError** : BOOL;
- **ipResultMessage** : I_TcMessage;
- **bBusy** : BOOL;
- **fControlGearExternalSupplyVoltageFrequency** : BYTE;

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable **ipResultMessage**. The output is set to FALSE again as soon as **bBusy** switches to TRUE.

**ipResultMessage:** Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after **bBusy** has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**fControlGearExternalSupplyVoltageFrequency:** Frequency of the external supply voltage [Hz] (0…253).
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### 3.1.2.8.8  FB_DALI253QueryControlGearOperatingTime

![FB_DALI253QueryControlGearOperatingTime](image)

The function block reads the operating time of the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nControlGearOperatingTime**: Operating time [s] (0…4294967293).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the output current as a percentage of the nominal current from the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input \( nAddress \) contains a short address (0…63) or a group address (0…15). Input \( nAddress \) has no meaning if a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( \text{E\_DALIAddressType} \) [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( \text{E\_DALICommandPriority} \) [549]).

\( e\text{CommandPriority} \) has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( \text{ipResultMessage} \). The output is set to FALSE again as soon as \( b\text{Busy} \) switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \( b\text{Busy} \) has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nControlGearOutputCurrentPercent**: Percentage output current [%] (0…100).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block indicates whether the limit value of the output power of the DALI control gear was exceeded.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>bControlGearOutputPowerLimitation</td>
<td>The output is set as soon as the output power of the DALI control gear is higher than the set limit value.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.11 FB_DALI253QueryControlGearOutputPowerLimitationCounter

The function block reads the counter of the number of exceedances of the output power limit value at the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

ℹ️ **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nControlGearOutputPowerLimitationCounter</td>
<td>Number of times the output power was exceeded (0…253).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block outputs the status of controlGearFailure [101] at the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Var Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType [547]`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority [549]`).

  `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Var Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bControlGearOverallFailureCondition</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bControlGearOverallFailureCondition**: The output is set as soon as an error has been detected in the DALI control gear.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.13  FB_DALI253QueryControlGearOverallFailureConditionCounter

The function block reads the counter of the error events that have occurred in the DALI control gear.

**VAR_INPUT**

```plaintext
bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
```

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI control gear or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

*eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

```plaintext
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nControlGearOverallFailureConditionCounter : BYTE;
```

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block. The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nControlGearOverallFailureConditionCounter:** Number of errors detected (0…253).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
### FB_DALI253QueryControlGearPowerFactor

The function block outputs the power factor of the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

  - *eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **fControlGearPowerFactor**: Power factor (0.0…1.0).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.15 FB_DALI253QueryControlGearStartCounter

The function block outputs the number of starts of the DALI control gear. The value is incremented after switching on the external power supply.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

- **nAddress**: Address of a DALI control gear or a DALI group.

- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).

- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

  **VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nControlGearStartCounter</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

- **nControlGearStartCounter**: Number of starts of the DALI control gear (0…16777213).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.16 FB_DALI253QueryControlGearTemperature

The function block outputs the internal temperature of the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see [E_DALIAddressType](#547)).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see [E_DALICommandPriority](#549)).

*eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nControlGearTemperature</td>
<td>INT</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see [error evaluation](#574)) that can be used to obtain detailed information about the processing of the function block (see [runtime messages](#561)). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nControlGearTemperature**: Internal temperature [°C] (-60…193).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block indicates whether the internal temperature is higher than the specified limit value in the DALI control gear.

### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

*Note: `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.*

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bControlGearThermalDerating**: The output is set as soon as the internally measured temperature of the DALI control gear is higher than the specified limit value.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.18 FB_DALI253QueryControlGearThermalDeratingCounter

The function block reads the counter that counts the number of times the temperature limit value inside the DALI control gear is exceeded.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nControlGearThermalDeratingCounter : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nControlGearThermalDeratingCounter: Number of times the internal temperature was exceeded (0…253).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.19 FB_DALI253QueryControlGearThermalShutdown

The function block indicates whether there is a thermal shutdown of the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bControlGearThermalShutdown : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

bControlGearThermalShutdown: The output is set as soon as the condition for a thermal shutdown of the DALI control gear is satisfied.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the counter of the number of thermal shutdowns of the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI control gear or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nControlGearThermalShutdownCounter</td>
<td>BYTE</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nControlGearThermalShutdownCounter**: Number of thermal shutdowns (0…253).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.21  
**FB_DALI253QueryInternalControlGearReferenceTemperature**

The function block outputs the internal reference temperature of the DALI control gear.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nInternalControlGearReferenceTemperature</td>
<td>INT;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nInternalControlGearReferenceTemperature**: Internal reference temperature [°C] (-60...193).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.22  FB_DALI253QueryLightSourceCurrent

The function block reads the current consumption of the light source.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]). eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>fLightSourceCurrent</td>
<td>LREAL</td>
<td>Current consumption [A] (0.0…65.533).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.23 FB_DALI253QueryLightSourceOnTime

The function block reads the non-resettable operating hour counter of the light source. If the light source is switched on, the non-resettable operating hour counter is incremented by one every second.

The non-resettable operating hour counter cannot be reset or changed.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

\[ \text{eCommandPriority} \text{ has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.} \]

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nLightSourceOnTime : UDINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nLightSourceOnTime: Current value of the non-resettable operating hour counter [s] (0...4294967293).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

TE1000 Version: 1.10
The function block reads the resettable operating hour counter of the light source. If the light source is switched on, the resettable operating hour counter is incremented by one every second. The resettable operating hour counter can be changed using the function block FB_DALI253SetLightSourceOnTimeResettable [284].

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nLightSourceOnTimeResettable**: Current value of the resettable operating hour counter of the light source [s] (0...4294967293).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.25  FB_DALI253QueryLightSourceOpenCircuit

The function block indicates whether there is an open circuit to the light source.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see *E_DALIAddressType*).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see *E_DALICommandPriority*).

  *eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bLightSourceOpenCircuit**: The output is set as soon as an open circuit to the light source is detected.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
FB_DALI253QueryLightSourceOpenCircuitCounter

The function block reads the counter of the number of open circuits to the light source.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nLightSourceOpenCircuitCounter**: Number of open circuits (0…253).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.27  FB_DALI253QueryLightSourceOverallFailureCondition

The function block indicates the status of lampFailure [101] on the DALI control gear.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
bLightSourceOverallFailureCondition : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nLightSourceOverallFailureCondition: The output is set as soon as a light source error has been detected in the DALI control gear.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the counter of the light source error events.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

  - **eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nLightSourceOverallFailureConditionCounter**: Number of errors detected (0…253).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.29  FB_DALI253QueryLightSourceShortCircuit

The function block indicates whether there is a short-circuit at the light source.

### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

> `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bLightSourceShortCircuit**: The output is set as soon as a short circuit is detected at the light source.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the counter that counts the number of light source short-circuits.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nLightSourceShortCircuitCounter</td>
<td>Number of short-circuits that have occurred (0…253).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.31  **FB_DALI253QueryLightSourceStartCounter**

The function block reads the non-resettable counter for the switching on of the light source. The non-resettable counter is incremented by one each time the light source changes from the Off state to the On state. The non-resettable counter cannot be reset or changed.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).

  *Note*: `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nLightSourceStartCounter**: Current value of the non-resettable counter for the switching on of the light source (0…16777213).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block reads the resettable counter for the switching on of the light source. The resettable counter is incremented by one each time the light source changes from the Off state to the On state. The resettable counter is writeable using the function block FB_DALI253SetLightSourceStartCounterResettable.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
<td>Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nLightSourceStartCounterResettable</td>
<td>UDINT</td>
<td>Current value of the resettable counter for the switching on of the light source (0...16777213).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.33  FB_DALI253QueryLightSourceTemperature

The function block indicates the temperature of the light source.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nLightSourceTemperature</td>
<td>Temperature of the light source [°C] (-60…193).</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
### 3.1.2.8.34 FB_DALI253QueryLightSourceThermalDerating

The function block indicates whether the light source temperature is higher than the specified limit value.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…15). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see *E_DALIAddressType* [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see *E_DALICommandPriority* [549]).
  
  *eCommandPriority* has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bLightSourceThermalDerating**: The output is set as soon as the temperature measured at the light source is higher than the specified limit value.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.35 FB_DALI253QueryLightSourceThermalDeratingCounter

The function block reads the counter that counts the number of times the light source temperature exceeded the limit value.

VAR_INPUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI control gear or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
<tr>
<td>nLightSourceThermalDeratingCounter</td>
<td>Number of times the light source temperature was exceeded (0…253).</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.36    FB_DALI253QueryLightSourceThermalShutdown

The function block indicates whether there is a thermal shutdown on the DALI control gear due to the light source.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

  eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bLightSourceThermalShutdown</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **bLightSourceThermalShutdown**: The output is set as soon as the condition for a thermal shutdown of the light source is satisfied.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
### 3.1.2.8.37 FB_DALI253QueryLightSourceThermalShutdownCounter

The function block reads the counter that counts the number of thermal shutdowns of the light source.

#### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

**eCommandPriority** has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

#### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nLightSourceThermalShutdownCounter**: Number of thermal shutdowns (0…253).

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.38   FB_DALI253QueryLightSourceVoltage

The function block reads the output voltage at the DALI control gear.

VAR_INPUT

| bStart   | BOOL;                     |
| nAddress | BYTE;                     |
| eAddressType | E_DALIAddressType := E_DALIAddressType.Short; |
| eCommandPriority | E_DALICommandPriority := E_DALICommandPriority.MiddleLow; |

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

\[ eCommandPriority \] has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

| bError  | BOOL;             |
| ipResultMessage | I_TcMessage; |
| bBusy   | BOOL;             |
| fLightSourceVoltage | LREAL; |

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

fLightSourceVoltage: Output voltage [V] (0.0…6553.3).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block indicates the estimated average lifetime of the complete luminaire, including all individual components.

The function block FB_DALI253SetRatedMedianUsefulLifeOfLuminaire writes the value to the DALI control gear.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI control gear or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
- **nRatedMedianUsefulLifeOfLuminaire**: Estimated average service life [1000 h]. The value lies within the range 0…253 as a multiple of the step size of 1000 h.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block indicates the average number of possible starts of the complete luminaire, including all individual components.

The function block FB_DALI253SetRatedMedianUsefulLightSourceStarts [p. 287] writes the value to the DALI control gear.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI control gear or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [p. 547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [p. 549]).

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nRatedMedianUsefulLightSourceStarts</td>
<td>UINT</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [p. 574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [p. 561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nRatedMedianUsefulLightSourceStarts: Number of average possible starts [100]. The value lies within the range 0...65533 as a multiple of the step size of 100.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
### FB_DALI253SetInternalControlGearReferenceTemperature

The function block writes the internal reference temperature to the DALI control gear. The function block `FB_DALI253QueryInternalControlGearReferenceTemperature` reads the value from the DALI control gear.

#### VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>Address of a DALI device or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType</td>
<td>Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).</td>
</tr>
<tr>
<td>nInternalControlGearReferenceTemperature</td>
<td>INT</td>
<td>Internal reference temperature [°C] (-60...193).</td>
</tr>
</tbody>
</table>

#### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.42  FB_DALI253SetLightSourceOnTimeResettable

The function block sets the resettable operating hour counter of the light source to any desired value. If the light source is switched on, the resettable operating hour counter is incremented by one every second.

The resettable operating hour counter can be read using the function block FB_DALI253QueryLightSourceOnTimeResettable [266].

VAR_INPUT

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nLightSourceOnTimeResettable</td>
<td>UDINT;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

   eCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nLightSourceOnTimeResettable: New value of the resettable operating hour counter [s] (0…4294967293).

VAR_OUTPUT

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.43  **FB_DALI253SetLightSourceStartCounterResettable**

The function block sets the resettable counter for the switching on of the light source to any desired value. The resettable counter is incremented by one each time the light source changes from the Off state to the On state.

The resettable counter can be read using the function block **FB_DALI253QueryLightSourceStartCounterResettable** [274].

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Var</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>Execution of the DALI commands is triggered via a positive edge at this input.</td>
</tr>
<tr>
<td>nAddress</td>
<td>Address of a DALI device or a DALI group.</td>
</tr>
<tr>
<td>eAddressType</td>
<td>Defines whether the input <code>nAddress</code> has a short address (0..63) or a group address (low, middle low, middle, middle high, high). Input <code>nAddress</code> has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see [E_DALIAddressType] [547]).</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see [E_DALICommandPriority] [549]).</td>
</tr>
<tr>
<td>nLightSourceStartCounterResettable</td>
<td>New value of the resettable counter for the switching on of the light source (0...16777213).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Var</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable <code>ipResultMessage</code>. The output is set to FALSE again as soon as <code>bBusy</code> switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after <code>bBusy</code> has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.8.44 FB_DALI253SetRatedMedianUsefulLifeOfLuminaire

The function block writes the estimated average lifetime of the complete luminaire to the DALI control gear.

The function block FB_DALI253QueryRatedMedianUsefulLifeOfLuminaire [281] reads the value from the DALI control gear.

**VAR_INPUT**

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nRatedMedianUsefulLifeOfLuminaire : BYTE;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...15). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

pCommandPriority has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.

nRatedMedianUsefulLifeOfLuminaire: Estimated average service life [1000 h]. The input value is between 0 and 253, as a multiple of the step size of 1000 h.

**VAR_OUTPUT**

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
The function block writes the average number of possible starts of the complete luminaire to the DALI control gear.

The function block `FB_DALI253QueryRatedMedianUsefulLightSourceStarts` [282] reads the value from the DALI control gear.

**VAR_INPUT**

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `nAddress`: Address of a DALI device or a DALI group.
- `eAddressType`: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…15). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType` [547]).
- `eCommandPriority`: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority` [549]).
  - `eCommandPriority` has no function with the KL6811. The command priorities are supported by the KL6821 from the firmware version BD.
- `nRatedMedianUsefulLightSourceStarts`: Number of average possible starts. The input value is between 0 and 65533, as a multiple of the step size of 100.

**VAR_OUTPUT**

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- `ipResultMessage`: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- `bBusy`: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.9 Part 301 (push button)

3.1.2.9.1 Introduction

IEC 62386-301 describes DALI control devices that are defined as push buttons.

DALI control devices according to IEC 62386-301 correspond to instance type 1. The type of an instance can be queried with the function block FB_DALI103QueryInstanceType.[137]

Input value

The current state (inputValue) of the push button can be queried with the function block FB_DALI103QueryInputValue.[132]. The value is 0xFF when the push button is actuated. The value is 0x00 when it is not actuated.

Timer

IEC 62386-301 defines specific timers with which the sending of the different events can be influenced.

The respective time is calculated from

\[ Time = T_{incr} \times Multiplier \]

<table>
<thead>
<tr>
<th>Time</th>
<th>Multiplier</th>
<th>Preset value of the multiplier</th>
<th>T_{incr} Incremental value</th>
<th>T_{default} default value</th>
<th>T_{min} lowest possible set value</th>
<th>T_{max} highest possible set value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{short}</td>
<td>tShort</td>
<td>Max (25, tShortMin)</td>
<td>20 ms</td>
<td>Max (25, tShortMin) * 20 ms</td>
<td>tShortMin * 20 ms</td>
<td>5100 ms</td>
</tr>
<tr>
<td>T_{double}</td>
<td>tDouble</td>
<td>0</td>
<td>20 ms</td>
<td>tDoubleMin * 20 ms</td>
<td>tDoubleMin * 20 ms</td>
<td>2000 ms</td>
</tr>
<tr>
<td>T_{repeat}</td>
<td>tRepeat</td>
<td>8</td>
<td>20 ms</td>
<td>tRepeatMin * 20 ms</td>
<td>160 ms</td>
<td>100 ms</td>
</tr>
<tr>
<td>T_{stuck}</td>
<td>tStuck</td>
<td>20</td>
<td>1 s</td>
<td>20 s</td>
<td>5 s</td>
<td>255 s</td>
</tr>
</tbody>
</table>

tShortMin: Vendor-specific (see tShortMin.[304]).

tDoubleMin: Vendor-specific (see tDoubleMin.[304]).

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

Event filter

DALI control devices are capable of sending different events. The event filter consists of an 8-bit value, where each individual bit defines whether an event is disabled or enabled. The event is enabled if the corresponding bit is set, otherwise it is disabled.

The function block FB_DALI103QueryEventFilter.[127] can be used to read the value of the event filter; the function block FB_DALI103SetEventFilter.[114] can be used to write to it.

The event filter (see eventFilter.[303]) must be configured accordingly in order to trigger time-controlled events.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Button Released</td>
<td>The push button was released.</td>
</tr>
<tr>
<td>1</td>
<td>Button Pressed</td>
<td>The push button was pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short Press</td>
<td>The push button was pressed and quickly released again (short button press).</td>
</tr>
<tr>
<td>3</td>
<td>Double Press</td>
<td>The push button was pressed, released and quickly pressed again (double button press).</td>
</tr>
<tr>
<td>4</td>
<td>Long Press Start</td>
<td>The push button was pressed without being released in between times (long button press).</td>
</tr>
<tr>
<td>5</td>
<td>Long Press Repeat</td>
<td>If the push button was pressed for a long period without being released in between times, this event is sent at regular intervals.</td>
</tr>
<tr>
<td>6</td>
<td>Long Press Stop</td>
<td>This event is sent if the push button is released after a long button press.</td>
</tr>
<tr>
<td>7</td>
<td>Button Free / Button Stuck</td>
<td>If the push button is pressed for a very long time, it is assumed that it has stuck and the Button Stuck event is sent. Button Free is sent when the push button is detected as not pressed again.</td>
</tr>
</tbody>
</table>

Release only those events that you need for the application. If too many events are sent, this can have a negative effect on the behavior of the application.

**Events**

With instance type 1 there is an option to define a series of different events with the help of the event filter. The following table lists the possible events bit by bit.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block `FB_DALI_ToPushButtonState` can be used to convert the state of a push button into a BOOL variable.

**Input Notification event**

The application controller is informed of all events with an Input Notification. Further information on the structure of the Input Notification can be found in the description of the function block `FB_DALIGetInputNotification`, with which the events are also received.
**Short Press event**

The *Short Press* event is triggered when the push button is released again within the time $T_{\text{short}}$ and not pressed again for at least the time $T_{\text{double}}$. If the push button is pressed again within the time $T_{\text{double}}$, a new *Short Press* event can be triggered.

**Double Press event**

The *Double Press* event is triggered when the push button is released within the time $T_{\text{short}}$ and pressed again before expiry of the time $T_{\text{double}}$.

**Long Press Start, Long Press Repeat and Long Press Stop events**

The *Long Press* event is triggered when the push button still remains pressed after expiry of the time $T_{\text{short}}$. The time $T_{\text{repeat}}$ subsequently starts. The *Long Press Repeat* event is triggered if this time has expired. If the push button remains pressed, the time $T_{\text{repeat}}$ starts again and the *Long Press Repeat* event is triggered after each expiry.
The *Long Press Stop* event is triggered by releasing the push button.

*Button Free and Button Stuck events*

The *Button Stuck* event is triggered as soon as the push button was pressed for the time $T_{stuck}$ without being released in between times. The *Button Free* event occurs immediately if the push button is released after the *Button Stuck* event.

The minimum set value for this time is 5 s; the default value is 20 s (see table above).

Note that the *Long Press* event does not temporally overlap the *Button Stuck* event, therefore the value should not be set too small.

### 3.1.2.9.2 FB_DALI301QueryDoubleTimer

The function block reads the variable $t_{Double}$.

The value is read in steps of 20 ms. The maximum permissible value is 2000 ms or 100. The value 0 disables the timer. The smallest permitted value can be queried with the function block FB_DALI301QueryDoubleTimerMin.

---

**VAR_INPUT**

- `bStart`: Execution of the DALI commands is triggered via a positive edge at this input.
- `nAddress`: Address of a DALI device or a DALI group.
**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see [E_DALIAddressType](#547)).

**nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see [E_DALIInstanceAddressType](#554)).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see [E_DALICommandPriority](#549)).

- The command priorities are supported by the KL6821 from the firmware version BD.

---

**VAR_OUTPUT**

```plaintext
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nDouble : BYTE;
```

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see [Error evaluation](#574)) that can be used to obtain detailed information about the processing of the function block (see [Runtime messages](#561)). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has started. It remains active until all DALI commands have been processed.

**nDouble**: Value of the variable `tDouble` [20 ms]. The maximum output value is 100, as a multiple of the step size of 20 ms.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### 3.1.2.9.3 **FB_DALI301QueryDoubleTimerMin**

The function block reads the variable `tDoubleMin` [304].

The value is read in steps of 20 ms.

- The function block cannot be used when using the KL6811.
VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nDoubleMin : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nDoubleMin: Value of the variable tDoubleMin [20 ms].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.9.4 FB_DALI301QueryRepeatTimer
The function block reads the variable \texttt{tRepeat}. The value is read in steps of 20 ms. The permissible value range is 100 ms to 2000 ms. It is output in numbers from 5 to 100.

The function block cannot be used when using the KL6811.

\textbf{VAR\_INPUT}

\begin{itemize}
    \item \texttt{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.
    \item \texttt{nAddress}: Address of a DALI device or a DALI group.
    \item \texttt{eAddressType}: Defines whether the input \texttt{nAddress} contains a short address (0…63) or a group address (0…31). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E\_DALIAddressType}).
    \item \texttt{nInstanceAddress}: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
    \item \texttt{eInstanceAddressType}: Defines the access type to the required instance within the DALI control device (see \texttt{E\_DALIInstanceAddressType}).
    \item \texttt{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E\_DALICommandPriority}).
\end{itemize}

The command priorities are supported by the KL6821 from the firmware version BD.

\textbf{VAR\_OUTPUT}

\begin{itemize}
    \item \texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.
    \item \texttt{ipResultMessage}: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.
    \item \texttt{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
    \item \texttt{nRepeat}: Value of the variable \texttt{tRepeat} [20 ms]. The output value is between 5 and 100, as a multiple of the step size of 20 ms.
\end{itemize}

\textbf{Requirements}

\begin{tabular}{|l|l|}
    \hline
    Development environment & required PLC library \\
    \hline
    TwinCAT from v3.1.4022.29 & Tc3\_DALI from v3.1.4.0 \\
    \hline
\end{tabular}
3.1.2.9.5 FB_DALI301QueryShortTimer

The function block reads the variable \( tShort \). The value is read in steps of 20 ms. The maximum permissible value is 5100 ms, or 255. The smallest permitted value can be queried with the function block \( FB_DALI301QueryShortTimerMin \).

\[ \begin{align*}
\text{i} & \quad \text{The function block cannot be used when using the KL6811.}
\end{align*} \]

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input \( nAddress \) contains a short address (0..63) or a group address (0..31). Input \( nAddress \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \( E_DALIAddressType \)).
- **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
- **eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see \( E_DALIInstanceAddressType \)).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \( E_DALICommandPriority \)).

\[ \begin{align*}
\text{i} & \quad \text{The command priorities are supported by the KL6821 from the firmware version BD.}
\end{align*} \]

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nShort</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( ipResultMessage \). The output is set to FALSE again as soon as \( bBusy \) switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after \( bBusy \) has changed from TRUE to FALSE.
**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nShort:** Value of the variable tShort [20 ms]. The maximum output value is 255, as a multiple of the step size of 20 ms.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### 3.1.2.9.6 FB_DALI301QueryShortTimerMin

The function block reads the variable tShortMin [\ref{304}].

The value is read in steps of 20 ms.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

- **bStart** : BOOL;
- **nAddress** : BYTE;
- **eAddressType** : E_DALIAddressType := E_DALIAddressType.Short;
- **nInstanceAddress** : BYTE := 0;
- **eInstanceAddressType** : E_DALIInstanceAddressType := E_DALIInstanceAddressType_InstanceNumber;
- **eCommandPriority** : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [\ref{547}]).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [\ref{554}]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [\ref{549}]).

The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

bError               : BOOL;
ipResultMessage     : I_TcMessage;
bBusy               : BOOL;
nShortMin           : BYTE;

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nShortMin:** Value of the variable `tShortMin` [20 ms].

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.9.7  **FB_DALI301QueryStuckTimer**

The function block reads the variable `tStuck` [304].

The value is read in steps of 1 s. The permissible value range is 5 s to 255 s.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart                  : BOOL;
nAddress                : BYTE;
eAddressType            : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress       : BYTE := 0;
eInstanceAddressType   : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority       : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \[E_DALICommandPriority] 549)).

The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError         : BOOL;</td>
<td></td>
</tr>
<tr>
<td>ipResultMessage: I_TcMessage;</td>
<td></td>
</tr>
<tr>
<td>bBusy          : BOOL;</td>
<td></td>
</tr>
<tr>
<td>nStuck         : BYTE;</td>
<td></td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage**: Interface pointer (see [Error evaluation] 574)) that can be used to obtain detailed information about the processing of the function block (see [Runtime messages] 561)). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nStuck**: Value of the variable tStuck [s].

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**3.1.2.9.8 FB_DALI301SetDoubleTimer**

The function block stores the value of nDouble in the variable tDouble 304] of the DALI control device.

The value is written in steps of 20 ms. The maximum permissible value is 2000 ms, or 100. The value 0 disables the timer. The smallest permitted value can be queried with the function block FB_DALI301QueryDoubleTimerMin 292].

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart         : BOOL;</td>
<td></td>
</tr>
<tr>
<td>nAddress       : BYTE;</td>
<td></td>
</tr>
<tr>
<td>eAddressType   : E_DALIAddressType ::= E_DALIAddressType.Short;</td>
<td></td>
</tr>
<tr>
<td>nInstanceAddress : BYTE := 0;</td>
<td></td>
</tr>
<tr>
<td>eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
<td></td>
</tr>
<tr>
<td>eCommandPriority : E_DALICommandPriority ::= E_DALICommandPriority.MiddleLow;</td>
<td></td>
</tr>
<tr>
<td>nDouble        : BYTE := 0;</td>
<td></td>
</tr>
</tbody>
</table>

The function block cannot be used when using the KL6811.
bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nDouble: Value of the variable tDouble [20 ms]. The maximum input value is 100, as a multiple of the step size of 20 ms.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.9.9 FB_DALI301SetRepeatTimer

The function block stores the value of nRepeat in the variable tRepeat [304] of the DALI control device.

The value is written in steps of 20 ms. The permissible value range is 100 ms to 2000 ms, i.e. from 5 to 100.
The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType_InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nRepeat : BYTE := 8;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nRepeat: Value of the variable tRepeat [20 ms]. The input value is between 2 and 100, as a multiple of the step size of 20 ms.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
3.1.2.9.10  FB_DALI301SetShortTimer

The function block stores the value of nShort in the variable tShort of the DALI control device.

The value is written in steps of 20 ms. The maximum permissible value is 5100 ms, or 255. The smallest permitted value can be queried with the function block FB_DALI301QueryShortTimerMin.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nShort : BYTE := 25;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority).

The command priorities are supported by the KL6821 from the firmware version BD.

nShort: Value of the variable tShort [20 ms]. The maximum input value is 255, as a multiple of the step size of 20 ms.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.9.11 FB_DALI301SetStuckTimer

The function block stores the value of nStuck in the variable tStuck [304] of the DALI control device. The value is read in steps of 1 s. The permissible value range is 5 s to 255 s.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nStuck</td>
<td>BYTE := 20;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nStuck: Value of the variable tStuck [s] (5...255 s).
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.9.12 Variables

Each instance of a DALI control device has a certain number of variables (parameters). These variables are used to configure the instance and thus influence its behavior. The values of the variables are stored in the respective DALI control device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>resolution [303]</td>
<td>No change</td>
<td>1</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>eventFilter [303]</td>
<td>2#1111_0100</td>
<td>2#XXXX_XXXX</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tShort [303]</td>
<td>Max(25, ShortMin)</td>
<td>tShortMin…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tShortMin [304]</td>
<td>No change</td>
<td>10…255</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>tDouble [304]</td>
<td>0</td>
<td>0, tDoubleMin…100</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tDoubleMin [304]</td>
<td>No change</td>
<td>10…100</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>tRepeat [304]</td>
<td>8</td>
<td>5…100</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tStuck [304]</td>
<td>20</td>
<td>5…255</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

resolution

Manufacturer-dependent value for the resolution of the input values.

eventFilter

Specific events for each instance are activated or deactivated with the event filter (see Introduction [288]). If the respective bit is set, the corresponding event is active.

The function block FB_DALI103QueryEventFilter [127] can be used to read the value of the event filter; the function block FB_DALI103SetEventFilter [114] can be used to write to it.

tShort

The variable defines the time value for a short keystroke. After the time has elapsed, a long keystroke is detected.

The value must be >= the value tShortMin. If tShortMin is < 25, then 25 is automatically adopted. According to the DALI standard, 25 (500 ms) is the physical minimum.
The value is processed in steps of 20 ms.

The function block FB_DALI301QueryShortTimer [295] can be used to query the value, the function block FB_DALI301SetShortTimer [301] can be used to specify it.

**tShortMin**

The variable tShortMin is a minimum value specified by the manufacturer for detecting a short keystroke.

The value can be queried with the function block FB_DALI301QueryShortTimerMin [296].

**tDouble**

The variable tDouble represents the time interval in which a repeated short keystroke must occur to be recognized as a double keystroke.

If the value is 0, the double keystroke event is disabled.

The value must be >= the value tDoubleMin.

100 is the maximum value of the variable. It corresponds to a time of 2000 ms for a step size of 20 ms.

The function block FB_DALI301QueryDoubleTimer [291] can be used to query the value, the function block FB_DALI301SetDoubleTimer [298] can be used to specify it.

**tDoubleMin**

The variable tDoubleMin is a minimum value specified by the manufacturer for detecting a double keystroke.

The value can be queried with the function block FB_DALI301QueryDoubleTimerMin [292].

**tRepeat**

The variable tRepeat specifies the interval for repeating an event when a long keystroke is executed.

The value is given in steps of 20 ms. The permissible value range is 5 (100 ms) to 100 (2000 ms).

The function block FB_DALI301QueryRepeatTimer [293] can be used to query the value, the function block FB_DALI301SetRepeatTimer [299] can be used to specify it.

**tStuck**

The variable tStuck indicates the time after a keystroke from which the key is considered to be stuck or defective.

The value is read in steps of 1 s. The permissible value range is 5 s to 255 s.

The function block FB_DALI301QueryStuckTimer [297] can be used to query the value, the function block FB_DALI301SetStuckTimer [302] can be used to specify it.

### 3.1.2.10 Part 302 (absolute input device)

#### 3.1.2.10.1 Introduction

IEC 62386-302 describes DALI control devices that are defined as absolute input device.

DALI control devices according to IEC 62386-302 correspond to instance type 2. The type of an instance can be queried with the function block FB_DALI103QueryInstanceType [137].
Input value

The current state (inputValue) of the absolute input devices can be queried via the function block FB_DALI103QueryInputValue. The variable can assume different values:

<table>
<thead>
<tr>
<th>inputValue</th>
<th>Slide resistor</th>
<th>Analog input</th>
<th>Switch or binary input</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Minimum position</td>
<td>Lower limit value</td>
<td>Opened</td>
</tr>
<tr>
<td>0x01, 2&lt;sup&gt;resolution - 2&lt;/sup&gt;</td>
<td>Linear position display between the minimum and maximum position (with a resolution &gt;2)</td>
<td>Linear display of the value within the upper and lower limits.</td>
<td>Closed</td>
</tr>
<tr>
<td>2&lt;sup&gt;resolution - 1&lt;/sup&gt;</td>
<td>Maximum position</td>
<td>Upper limit value</td>
<td>Closed</td>
</tr>
</tbody>
</table>

Depending on the connected DALI control device, the inputValue can assume values that depend on the resolution (see Samples).

Timer

IEC 62386 Part 302 defines specific timers with which the sending of the different events can be influenced. The respective time is calculated from

\[
Time = T_{\text{incr}} \times \text{Multiplier}
\]

<table>
<thead>
<tr>
<th>Time</th>
<th>Multiplier</th>
<th>Preset value of the multiplier</th>
<th>T\text{incr} Incremental value</th>
<th>T\text{default} default value</th>
<th>T\text{min} lowest possible set value</th>
<th>T\text{max} highest possible set value</th>
</tr>
</thead>
<tbody>
<tr>
<td>t\text{deadtime}</td>
<td>t\text{Deadtime}</td>
<td>2</td>
<td>50 ms</td>
<td>100 ms</td>
<td>0 s</td>
<td>12.75 s</td>
</tr>
<tr>
<td>t\text{report}</td>
<td>t\text{Report}</td>
<td>0</td>
<td>1 s</td>
<td>Vendor-specific</td>
<td>1 s</td>
<td>4 min 15 s</td>
</tr>
</tbody>
</table>

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

Event filter

DALI control devices are capable of sending different events. The event filter consists of an 8-bit value, where each individual bit defines whether an event is blocked or released. The event is released if the corresponding bit is set, otherwise it is blocked.

The function block FB_DALI103QueryEventFilter can be used to read the value of the event filter; the function block FB_DALI103SetEventFilter can be used to write to it.

The event filter (see eventFilter) must be configured accordingly in order to trigger time-controlled events. The default setting of the event filter is shown in the table further below.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Position Report</td>
<td><code>inputValue</code> has changed or the Report Timer has expired.</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Input Notification event**

The application controller is informed of all events with an *Input Notification*. Further information on the structure of the *Input Notification* can be found in the description of the function block `FB_DALIGetInputNotification [543]`, with which the events are also received.

**Deadtime timer**

Several events may not be sent from an instance at the same time. To avoid this a dead time ($T_{deadtime}$) can be activated (see `tDeadtime [312]`).

If the Deadtime Timer is active, an instance should not send any further events until the timer $T_{deadtime}$ has expired. The timer is restarted after the sending of an event.

If a new event occurs before the current event is sent, then the new event replaces the existing one. This can happen if the bus is not available or if the timer $T_{deadtime}$ is in use.

If $T_{report} < T_{deadtime}$, then $T_{report}$ should be equal to $T_{deadtime}$ regardless of the value that $T_{report}$ previously had.

### 3.1.2.10.2 FB_DALI302QueryDeadtimeTimer

The function block reads the variable `tDeadtime [312]`.

The value is read in steps of 50 ms. The permissible value range is 0 s to 12.75 s. It is output in numbers from 0 to 255.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

- `bStart`: BOOL;
- `nAddress`: BYTE;
- `eAddressType`: E_DALIAddressType := E_DALIAddressType.Short;
- `nInstanceAddress`: BYTE := 0;
- `eInstanceAddressType`: E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
- `eCommandPriority`: E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

*bStart*: Execution of the DALI commands is triggered via a positive edge at this input.
**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E.DALIAddressType [547]`).

**nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see `E.DALIInstanceAddressType [554]`).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E.DALICommandPriority [549]`).

The command priorities are supported by the KL6821 from the firmware version BD.

### VAR OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nDeadtime</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nDeadtime**: Value of the variable tDeadtime [50 ms]. The output value is between 0 and 255, as a multiple of the step size of 50 ms.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### 3.1.2.10.3 FB_DALI302QueryReportTimer

The function block reads the variable `tReport [313]`.

The value is read in steps of 1 s. The permissible value range is 0 s to 4 min 15 s. It is output in numbers from 0 to 255.
The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nReport : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nReport: Value of the variable tReport [1 s]. The output value is between 0 and 255, as a multiple of the step size of 1 s.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.1.2.10.4 FB_DALI302QuerySwitch

The function block queries whether the instance represents a switch.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0...63) or a group address (0...31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

- The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bSwitch</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**bSwitch:** The output is set if the instance represents a switch.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.10.5 FB_DALI302SetDeadtimeTimer

The function block stores the value of `nDeadtime` in the variable `tDeadtime[50 ms]` of the DALI control device.

The value is written in steps of 50 ms. The permissible value range is 0 s to 12.75 s. It is entered in numbers from 0 to 255.

- **The function block cannot be used when using the KL6811.**

VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType[547]`).
- **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
- **eInstanceAddressType**: (see `E_DALIInstanceAddressType[554]`)
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority[549]`).

- **The command priorities are supported by the KL6821 from the firmware version BD.**

- **nDeadtime**: Value of the variable `tDeadtime [50 ms]`. The input value is between 0 and 255, as a multiple of the step size of 50 ms.

VAR_OUTPUT

- **bError**: Bool
- **ipResultMessage**: I_TcMessage
- **bBusy**: Bool

The function block cannot be used when using the KL6811.
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.1.2.10.6 FB_DALI302SetReportTimer

The function block stores the value of nReport in the variable tReport [313] of the DALI control device. The value is written in steps of 1 s. The permissible value range is 0 s to 4 min 15 s. It is entered in numbers from 0 to 255.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart : BOOL;</td>
</tr>
<tr>
<td>nAddress : BYTE;</td>
</tr>
<tr>
<td>eAddressType : E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress : BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority : E.DALICommandPriority := E.DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nReport : BYTE := 0;</td>
</tr>
</tbody>
</table>

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: (see E_DALIInstanceAddressType [554])

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
The command priorities are supported by the KL6821 from the firmware version BD.

\[ n\text{Report} \]\( : \) Value of the variable \( t\text{Report} \) \([1 \text{ s}]\). The input value is between 0 and 255, as a multiple of the step size of 1 s.

\[ \text{VAR_OUTPUT} \]

\begin{verbatim}
VAR_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
\end{verbatim}

\[ b\text{Error} \]\( : \) This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( ip\text{ResultMessage} \). The output is set to FALSE again as soon as \( b\text{Busy} \) switches to TRUE.

\[ ip\text{ResultMessage} \]\( : \) Interface pointer (see Error evaluation \([574]\)) that can be used to obtain detailed information about the processing of the function block (see Runtime messages \([561]\)). The interface pointer is valid after \( b\text{Busy} \) has changed from TRUE to FALSE.

\[ b\text{Busy} \]\( : \) The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

\[ \text{Requirements} \]

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### 3.1.2.10.7 Variables

Each instance of a DALI control device has a certain number of variables (parameters). These variables are used to configure the instance and thus influence its behavior. The values of the variables are stored in the respective DALI control device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>evenFilter ([312])</td>
<td>2#0000_0001</td>
<td>2#0000_000X</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tDeadtime ([312])</td>
<td>2</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tReport ([313])</td>
<td>0</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{eventFilter} \]

Specific events for each instance are enabled or disabled with the event filter (see Introduction \([305]\)). If the respective bit is set, the corresponding event is active.

The function block \( \text{FB\_DALI103QueryEventFilter} \) \([127]\) can be used to read the value of the event filter; the function block \( \text{FB\_DALI103SetEventFilter} \) \([114]\) can be used to write to it.

\[ \text{tDeadtime} \]

The variable \( t\text{Deadtime} \) defines the time value for a dead time in which no new event is sent until the dead time has expired. If this timer is active, it is restarted after each transmitted event.

If the value is 0, the dead time event is disabled.

The value is given in steps of 50 ms. The permissible value range is 0 to 255 (12.75 s).

The function block \( \text{FB\_DALI302QueryDeadtimeTimer} \) \([306]\) can be used to query the value, the function block \( \text{FB\_DALI302SetDeadtimeTimer} \) \([310]\) can be used to specify it.
tReport

The variable \( tReport \) specifies the report time of an absolute input device after the occurrence of an event. On expiry of the time specified in the variable \( tReport \), an event is triggered and the time is restarted. If an event was triggered before the expiry of the time \( tReport \), the time starts over.

The value is given in steps of 1 s. The permissible value range is 0 (0 s) to 255 (4 min 15 s).

The function block FB_DALI302QueryReportTimer \( \Rightarrow 307 \) can be used to query the value, the function block FB_DALI302SetReportTimer \( \Rightarrow 311 \) can be used to specify it.

3.1.2.11 Part 303 (occupancy sensor)

3.1.2.11.1 Introduction

IEC 62386-303 describes DALI control devices that are defined as occupancy sensors. DALI control devices according to IEC 62386-303 correspond to instance type 3. The type of an instance can be queried with the function block FB_DALI103QueryInstanceType \( \Rightarrow 137 \).

Input value

The current state (inputValue) of the light sensor can be queried via the function block FB_DALI103QueryInputValue \( \Rightarrow 132 \).

The variable can assume four values:

<table>
<thead>
<tr>
<th>inputValue</th>
<th>Occupancy</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>No+</td>
<td>No+</td>
</tr>
<tr>
<td>0x55</td>
<td>No+</td>
<td>Yes</td>
</tr>
<tr>
<td>0xAA</td>
<td>Yes</td>
<td>No+</td>
</tr>
<tr>
<td>0xFF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

For simplification the graphics below only show whether an area is occupied or vacant.

Timer

IEC 62386 Part 303 defines specific timers with which the sending of the different events can be influenced.

The respective time is calculated from

\[
Time = T_{\text{incr}} \times \text{Multiplier}
\]

<table>
<thead>
<tr>
<th>Time</th>
<th>Multiplier</th>
<th>Preset value of the multiplier</th>
<th>( T_{\text{incr}} ) incremental value</th>
<th>( T_{\text{default}} ) default value</th>
<th>( T_{\text{min}} ) lowest possible set value</th>
<th>( T_{\text{max}} ) highest possible set value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{\text{deadtime}} )</td>
<td>( t_{\text{deadtime}} )</td>
<td>2</td>
<td>50 ms</td>
<td>100 ms</td>
<td>0 s</td>
<td>12.75 s</td>
</tr>
<tr>
<td>( T_{\text{hold}} )</td>
<td>( t_{\text{hold}} )</td>
<td>90</td>
<td>10 s</td>
<td>15 min</td>
<td>1 s</td>
<td>42 min 20 s</td>
</tr>
<tr>
<td>( T_{\text{report}} )</td>
<td>( t_{\text{report}} )</td>
<td>20</td>
<td>1 s</td>
<td>20 s</td>
<td>1 s</td>
<td>4 min 15 s</td>
</tr>
</tbody>
</table>

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.
**Event filter**

DALI control devices are capable of sending different events. The event filter consists of an 8-bit value, where each individual bit defines whether an event is blocked or released. The event is released if the corresponding bit is set, otherwise it is blocked.

The function block `FB_DALI103QueryEventFilter` can be used to read the value of the event filter; the function block `FB_DALI103SetEventFilter` can be used to write to it.

The event filter (see `eventFilter`) must be configured accordingly in order to trigger time-controlled events. The default setting of the event filter is shown in the table further below.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Occupied</td>
<td>Occupancy detection</td>
</tr>
<tr>
<td>1</td>
<td>Vacant</td>
<td>Area vacant detection</td>
</tr>
<tr>
<td>2</td>
<td>Repeat</td>
<td>Event repeat</td>
</tr>
<tr>
<td>3</td>
<td>Movement</td>
<td>Movement detection</td>
</tr>
<tr>
<td>4</td>
<td>No Movement</td>
<td>No Movement detection</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Release only those events that you need for the application. If too many events are sent, this can have a negative effect on the behavior of the application.

**Events**

With instance type 3 it is possible with the help of the event filter to define various possible combinations of events that may occur together. The possible events (`Event Name` column) and their binary representation (`Event Information` column) are listed in the following table.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Event Information</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Movement</td>
<td>00 0000 XXX0b</td>
<td>No movement detected.</td>
</tr>
<tr>
<td>Movement</td>
<td>00 0000 XXX1b</td>
<td>Movement detected.</td>
</tr>
<tr>
<td>Vacant</td>
<td>00 0000 X00Xb</td>
<td>The area is not occupied.</td>
</tr>
<tr>
<td>Still Vacant</td>
<td>00 0000 X10Xb</td>
<td>The area is vacant over a lengthy period. The event occurs at regular intervals as long as the vacant condition is satisfied.</td>
</tr>
<tr>
<td>Occupied</td>
<td>00 0000 X01Xb</td>
<td>The area is occupied.</td>
</tr>
<tr>
<td>Still Occupied</td>
<td>00 0000 X11Xb</td>
<td>The area is occupied over a lengthy period. The event occurs at regular intervals as long as the occupied condition is satisfied.</td>
</tr>
<tr>
<td>Presence Sensor</td>
<td>00 0000 0XXXb</td>
<td>The current event was triggered by an occupancy sensor.</td>
</tr>
<tr>
<td>Movement Sensor</td>
<td>00 0000 1XXXb</td>
<td>The current event was triggered by a motion sensor.</td>
</tr>
</tbody>
</table>

The points marked with X can assume a value of either 0 or 1 and thus combine several events with one another.

**Input Notification event**

The application controller is informed of all events with an Input Notification. Further information on the structure of the Input Notification can be found in the description of the function block `FB_DALIGetInputNotification`, with which the events are also received.
Motion sensor

The instance detects exclusively whether there is movement in the room or not.
If no movement was detected within a certain time, the area is regarded as vacant.

Occupancy sensor

The instance detects occupancy of the room. Some devices detect occupancy and movement.
Further information can be found in IEC 62386 Part 303.

Events with HOLD TIMER

On expiry, the time $T_{\text{report}}$ triggers the "Repeat" event (bit 2 of the event filter) (see $t_{\text{Report}}$). The time then starts over. The previous event remains active during the time $T_{\text{report}}$. $T_{\text{report}}$ is restarted by a newly detected event.

$T_{\text{hold}}$ provides for a hold time following the detection of movement (see $t_{\text{Hold}}$). Not all devices support this variable.

Events with CANCEL HOLD TIMER

If the DALI device supports the hold timer, it can be deactivated via the command $FB_{\text{DALI303CancelHoldTimer}}$.
The hold timer was deactivated before the expiry of its set time. The **Vacant** und **No Movement** events are triggered immediately at this moment.

**Deadtime timer**

Several events may not be sent from an instance at the same time. To avoid this a dead time ($T_{\text{deadtime}}$) can be activated (see $t_{\text{Deadtime}}$ [327]).

If the Deadtime Timer is active, an instance should not send any further events until the timer $T_{\text{deadtime}}$ has expired. The timer is restarted after the sending of an event.

If a new event occurs before the current event is sent, then the new event replaces the existing one. This can happen if the bus is not available or if the timer $T_{\text{deadtime}}$ is in use.

If $T_{\text{report}} < T_{\text{deadtime}}$, then $T_{\text{report}}$ should be equal to $T_{\text{deadtime}}$ regardless of the value that $T_{\text{report}}$ previously had.

### 3.1.2.11.2 FB_DALI303CancelHoldTimer

![FB_DALI303CancelHoldTimer](image)

The function block ends the Hold Timer prematurely if the latter was implemented in the DALI control device.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input $nAddress$ contains a short address (0…63) or a group address (0…31). Input $nAddress$ has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see $E_{\text{DALIAddressType}}$ [547]).
- **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
- **eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see $E_{\text{DALIInstanceAddressType}}$ [554]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see $E_{\text{DALICommandPriority}}$ [549]).

The command priorities are supported by the KL6821 from the firmware version BD.
VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

- **ipResultMessage**: Interface pointer (see Error evaluation [5.74]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [5.61]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### 3.1.2.11.3 FB_DALI303CatchMovement

**Catching** is activated for the DALI control devices by this function block. If the DALI control device detects motion, an event is sent once and the **Catching** is reset.

The **Event Filter** [3.27] should be set so that the **Movement** event is disabled. This ensures that the event is only sent once as soon as **Catching** is active and movement is detected.

The function block cannot be used when using the KL6811.

VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…31). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [5.547]).
- **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
- **eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [5.554]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see **E_DALICommandPriority** [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

---

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable **ipResultMessage**. The output is set to FALSE again as soon as **bBusy** switches to TRUE.

**ipResultMessage**: Interface pointer (see **Error evaluation** [574]) that can be used to obtain detailed information about the processing of the function block (see **Runtime messages** [561]). The interface pointer is valid after **bBusy** has changed from TRUE to FALSE.

**bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

---

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

---

**3.1.2.11.4 FB_DALI303QueryCatching**

The function block queries whether **Catching** is active.

**Catching** can be activated by **FB_DALI303CatchMovement** [317].

The function block cannot be used when using the KL6811.

---

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input **nAddress** contains a short address (0...63) or a group address (0...31). Input **nAddress** has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see **E_DALIAddressType** [547]).
**nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

---

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

- **bCatching**: *Catching* is active.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### 3.1.2.11.5 FB_DALI303QueryDeadtimeTimer

The function block reads the variable **tDeadtime [327]**.

The value is read in steps of 50 ms. The permissible value range is 0 s to 12.75 s. It is output in numbers from 0 to 255.

The function block cannot be used when using the KL6811.

---

**VAR_INPUT**

- **bStart**: BOOL;
- **nAddress**: BYTE;
- **eAddressType**: E_DALIAddressType := E_DALIAddressType.Short;
- **eCommandPriority**: E_DALICommandPriority;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nDeadtime : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nDeadtime: Value of the variable tDeadtime [50 ms]. The output value is between 0 and 255, as a multiple of the step size of 50 ms.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.11.6   FB_DALI303QueryHoldTimer

The function block reads the variable tHold [327].
The value is read in steps of 10 s. The permissible value range is 1 s to 42.5 min. It is output in numbers from 0 to 254.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bStart</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType [547]`).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see `E_DALIInstanceAddressType [554]`).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority [549]`).

The command priorities are supported by the KL6821 from the firmware version BD.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>nHold</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see `Error evaluation [574]`) that can be used to obtain detailed information about the processing of the function block (see `Runtime messages [561]`). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nHold:** Value of the variable `tHold` [10 s]. The output value is between 1 s and 42.5 min, as a multiple of the step size of 10 s.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
### 3.1.2.11.7 FB_DALI303QueryReportTimer

The function block reads the variable `tReport`.

The value is read in steps of 1 s. The permissible value range is 0 s to 4 min 15 s. It is output in numbers from 0 to 255.

The function block cannot be used when using the KL6811.

#### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see `E_DALIAddressType`).
- **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
- **eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see `E_DALIInstanceAddressType`).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see `E_DALICommandPriority`).

The command priorities are supported by the KL6821 from the firmware version BD.

#### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see Runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
nReport: Value of the variable tReport [1 s]. The output value is between 0 and 255, as a multiple of the step size of 1 s.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.11.8 FB.DALI303SetDeadtimeTimer

The function block stores the value of nDeadtime in the variable tDeadtime [50 ms]. The input value is between 0 and 255, as a multiple of the step size of 50 ms.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType_InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nDeadtime : BYTE := 2;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nDeadtime: Value of the variable tDeadtime [50 ms]. The input value is between 0 and 255, as a multiple of the step size of 50 ms.
**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage:** Interface pointer (see *Error evaluation [574]*) that can be used to obtain detailed information about the processing of the function block (see *Runtime messages [561]*). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**3.1.2.11.9 FB_DALI303SetHoldTimer**

The function block stores the value of *nHold* in the variable *tHold [327]* of the DALI control device.

The value is written in steps of 10 s. The permissible value range is 1 s to 42.3 min. It is entered in numbers from 0 to 254. If the value 0 is passed, the variable *tHold* is set to 1 s.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>bStart</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>nAddress</td>
<td>BYTE;</td>
</tr>
<tr>
<td>eAddressType</td>
<td>E_DALIAddressType := E_DALIAddressType.Short;</td>
</tr>
<tr>
<td>nInstanceAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>eInstanceAddressType</td>
<td>E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;</td>
</tr>
<tr>
<td>eCommandPriority</td>
<td>E_DALICommandPriority := E_DALICommandPriority.MiddleLow;</td>
</tr>
<tr>
<td>nHold</td>
<td>BYTE := 90;</td>
</tr>
</tbody>
</table>

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress:** Address of a DALI device or a DALI group.

**eAddressType:** Defines whether the input *nAddress* contains a short address (0...63) or a group address (0...31). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see *E_DALIAddressType [547]*).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see *E_DALIInstanceAddressType [554]*).
**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see [E_DALICommandPriority](#549)).

The command priorities are supported by the KL6821 from the firmware version BD.

**nHold:** Value of the variable `tHold` [10 s]. The input value is between 0 and 254, as a multiple of the step size of 10 s.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see [Error evaluation](#574)) that can be used to obtain detailed information about the processing of the function block (see [Runtime messages](#561)). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### 3.1.2.11.10 FB_DALI303SetReportTimer

The function block stores the value of `nReport` in the variable `tReport` [327] of the DALI control device.

The value is written in steps of 1 s. The permissible value range is 0 s to 4 min 15 s. It is entered in numbers from 0 to 255.

**bStart:** Execution of the DALI commands is triggered via a positive edge at this input.
nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nReport: Value of the variable tReport [1 s]. The input value is between 0 and 255, as a multiple of the step size of 1 s.

VAR_OUTPUT

VAR_OUTPUT

bError : BOOL;
ipResultMessage: I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.11.11 Variables

Each instance of a DALI control device has a certain number of variables (parameters). These variables are used to configure the instance and thus influence its behavior. The values of the variables are stored in the respective DALI control device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables [327]</td>
<td>No change</td>
<td>2</td>
<td>1 byte</td>
<td>Read only</td>
</tr>
<tr>
<td>Variables [327]</td>
<td>2#0000_0011</td>
<td>2#000X_XXXX</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [327]</td>
<td>2</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [327]</td>
<td>90</td>
<td>0…254</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Variables [327]</td>
<td>20</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

X: not specified
resolution
Manufacturer-dependent value for the resolution of the input values.

eventFilter
Specific events for each instance are activated or deactivated with the event filter (see Introduction [314]).
If the respective bit is set, the corresponding event is active.

The function block FB_DALI103QueryEventFilter [127] can be used to read the value of the event filter; the function block FB_DALI103SetEventFilter [114] can be used to write to it.

tDeadtime
The variable defines the time value for a dead time in which no new event is sent until the dead time has elapsed. If this timer is active, it is restarted after each transmitted event.
If the value is 0, the dead time event is disabled.
The value is given in steps of 50 ms. The permissible value range is 0 to 255 (12.75 s).
The function block FB_DALI303QueryDeadtimeTimer [319] can be used to query the value, the function block FB_DALI303SetDeadtimeTimer [323] can be used to specify it.

The hold time \( t_{Hold} \) can be terminated prematurely with the function block FB_DALI303CancelHoldTimer [316].

tReport
The variable \( t_{Report} \) specifies the report time of an occupancy sensor after the occurrence of an event.
After the time specified in the variable \( t_{Report} \) has elapsed, the repeat event from the event filter is enabled. This keeps the occupied event active.
The value is given in steps of 1 s. The permissible value range is 0 (0 s) to 255 (4 min 15 s).
The function block FB_DALI303QueryReportTimer [322] can be used to query the value, the function block FB_DALI303SetReportTimer [325] can be used to specify it.

3.1.2.12 Part 304 (light sensors)

3.1.2.12.1 Introduction
IEC 62386-304 describes DALI control devices that are defined as brightness sensors.
DALI control devices according to IEC 62386-304 correspond to instance type 4. The type of an instance can be queried with the function block FB_DALI103QueryInstanceType [137].
Input value

The current state (inputValue) of the light sensor can be queried via the function block FB_DALI103QueryInputValue [132].

Timer

IEC 62386 Part 304 defines various timers for triggering device-specific events. The event for the instance type 4 in conjunction with the associated timers is explained below. The respective time is calculated from

\[ Time = T_{\text{incr}} \times \text{Multiplier} \]

<table>
<thead>
<tr>
<th>Time</th>
<th>Multiplier</th>
<th>Preset value of the multiplier</th>
<th>( T_{\text{incr}} ) incremental value</th>
<th>( T_{\text{default}} ) default value</th>
<th>( T_{\text{min}} ) lowest possible set value</th>
<th>( T_{\text{max}} ) highest possible set value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{\text{deadtime}} )</td>
<td>( t_{\text{deadtime}} )</td>
<td>30</td>
<td>50 ms</td>
<td>1.5 s</td>
<td>0 s</td>
<td>12.75 s</td>
</tr>
<tr>
<td>( T_{\text{report}} )</td>
<td>( t_{\text{report}} )</td>
<td>30</td>
<td>1 s</td>
<td>30 s</td>
<td>1 s</td>
<td>4 min 15 s</td>
</tr>
</tbody>
</table>

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

Event filter

DALI control devices are capable of sending different events. The event filter consists of an 8-bit value, where each individual bit defines whether an event is blocked or released. The event is released if the corresponding bit is set, otherwise it is blocked.

The function block FB_DALI103QueryEventFilter [127] can be used to read the value of the event filter; the function block FB_DALI103SetEventFilter [114] can be used to write to it.

The event filter (see eventFilter [340]) must be configured accordingly in order to trigger time-controlled events. The default setting of the event filter is shown in the table further below.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Illuminance level report</td>
<td>Illuminance event</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Input Notification event

The application controller is informed of all events with an Input Notification. Further information on the structure of the Input Notification can be found in the description of the function block FB_DALIGetInputNotification [543], with which the events are also received.

Hysteresis

To avoid too frequent, disruptive changes of the illuminance, it is possible with DALI devices with the instance type 4 to define a hysteresis.
A percentage value can be specified via the `hysteresis` variable for the calculation of a hysteresis band.

The function block `FB_DALI304QueryHysteresis` can be used to query the value, the function block `FB_DALI304SetHysteresis` can be used to specify it.

Since the percentage hysteresis can also lead to wide fluctuations at low illuminance, a minimum hysteresis can be specified via the variable `hysteresisMin`. The minimum hysteresis is an absolute value with a range from 0 to 255.

The reset value depends on the resolution of the input values specified by the vendor.

<table>
<thead>
<tr>
<th>resolution</th>
<th>Vendor-dependent reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>40</td>
</tr>
<tr>
<td>13</td>
<td>81</td>
</tr>
<tr>
<td>14</td>
<td>163</td>
</tr>
<tr>
<td>&gt; 15</td>
<td>255</td>
</tr>
</tbody>
</table>

The function block `FB_DALI304QueryHysteresisMin` can be used to query the value, the function block `FB_DALI304SetHysteresisMin` can be used to specify it manually.

**Illuminance Level Report event**

The illuminance event is triggered each time an `inputValue` lies outside of the hysteresis band.

As described above, the hysteresis band is calculated from the percentage hysteresis of the input value and the value of the minimum hysteresis. The upper and lower points of the hysteresis band are called `hysteresisBandHigh` and `hysteresisBandLow` respectively.

The illuminance event is generated when the `inputValue` is greater than the calculated value `hysteresisBandHigh` or smaller than `hysteresisBandLow`.

It is also generated when the time `T_report` has expired, irrespective of the current input value.
The example shows the change in the input value with the resulting hysteresis band and the report timer \( T_{\text{report}} \). The value for hysteresis is 10% and the value for hysteresisMin is 50.

The **Illuminance Level Report** event is generated at the measuring points 1, 2, 4, 5, 6, 8 and 10 through the change in the input value beyond the limits of the defined hysteresis band. No illuminance event is generated at the measuring points 3, 7 and 9 because the measured illuminance is within the hysteresis band and the timer \( T_{\text{report}} \) (see \( t\text{Report} \)) has not yet expired. On expiry of the timer \( T_{\text{report}} \) the event is generated at a later time (between the measuring points 3 and 4, 7 and 8, 9 and 10).

**Deadtime timer**

Several events may not be sent from an instance at the same time. To avoid this a dead time \( (T_{\text{deadtime}}) \) can be activated (see \( t\text{Deadtime} \)).

If the Deadtime Timer is active, an instance should not send any further events until the timer \( T_{\text{deadtime}} \) has expired. The timer is restarted after the sending of an event.

If a new event occurs before the current event is sent, then the new event replaces the existing one. This can happen if the bus is not available or if the timer \( T_{\text{deadtime}} \) is in use.

If \( T_{\text{report}} < T_{\text{deadtime}} \), then \( T_{\text{report}} \) should be equal to \( T_{\text{deadtime}} \) regardless of the value that \( T_{\text{report}} \) previously had.

### 3.1.2.12.2 FB_DALI304QueryDeadtimeTimer

The function block reads the variable \( t\text{Deadtime} \).

The value is read in steps of 50 ms. The permissible value range is 0 s to 12.75 s. It is output in numbers from 0 to 255.

The function block cannot be used when using the KL6811.

#### VAR_INPUT

- **bStart**: BOOL;
- **nAddress**: BYTE;
- **eAddressType**: E_DALIAddressType := E_DALIAddressType.Short;
- **nInstanceAddress**: BYTE := 0;
- **eInstanceAddressType**: E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
- **eCommandPriority**: E_DALICommandPriority := E_DALICommandPriority.MiddleLow;

**bStart**: Execution of the DALI commands is triggered via a positive edge at this input.

**nAddress**: Address of a DALI device or a DALI group.

**eAddressType**: Defines whether the input \( nAddress \) contains a short address (0…63) or a group address (0…31). Input \( nAddress \) has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType).

**nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType).

330  Version: 1.10  TE1000
eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nDeadtime : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nDeadtime: Value of the variable tDeadtime [50 ms]. The output value is between 0 and 255, as a multiple of the step size of 50 ms.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.12.3 FB_DALI304QueryHysteresis

The function block reads the variable hysteresis [341].

The value is stated in %. The permissible value range is between 0% and 25%.

The function block cannot be used when using the KL6811.

VAR_INPUT
bStart : BOOL;
nAddress : BYTE;
eAddressType := E DALIAddressType.Short;
nInstanceAddress := 0;
eInstanceAddressType := E DALIInstanceAddressType.InstanceNumber;
eCommandPriority := E DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.
**eAddressType:** Defines whether the input `nAddress` contains a short address (0…63) or a group address (0…31). Input `nAddress` has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see **E_DALIAddressType** [547]).

**nInstanceAddress:** Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

**eInstanceAddressType:** Defines the access type to the required instance within the DALI control device (see **E_DALIInstanceAddressType** [554]).

**eCommandPriority:** Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see **E_DALICommandPriority** [549]).

- The command priorities are supported by the KL6821 from the firmware version BD.

### VAR_OUTPUT

- `bError` : BOOL;
- `ipResultMessage` : I_TcMessage;
- `bBusy` : BOOL;
- `nHysteresis` : BYTE;

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bBusy:** The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**nHysteresis:** Value of the variable `hysteresis` in %.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### 3.1.2.12.4 FB_DALI304QueryHysteresisMin

The function block reads the variable `hysteresisMin` [340].

The permissible value range is between 0 and 255.

Since the hysteresis is given as a percentage, the absolute value of the hysteresis depends on the actual measured value. For very small measured values, the hysteresis is therefore also very small. This causes unnecessary events to be sent. For this reason, a minimum possible hysteresis can be set.
The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;

nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress := 0;
eInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority := E_DALICommandPriority.MiddleLow;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

VAR_OUTPUT

bError : BOOL;

ipResultMessage : I_TcMessage;
bBusy : BOOL;
nHysteresisMin := BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

nHysteresisMin: Value of the smallest possible hysteresis.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
### 3.1.2.12.5 FB.DALI304QueryReportTimer

The function block reads the variable \texttt{tReport} [340].

The value is read in steps of 1 s. The permissible value range is 0 s to 4 min 15 s. It is output in numbers from 0 to 255.

- **VAR_INPUT**
  - \texttt{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.
  - \texttt{nAddress}: Address of a DALI device or a DALI group.
  - \texttt{eAddressType}: Defines whether the input \texttt{nAddress} contains a short address (0...63) or a group address (0...31). Input \texttt{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \texttt{E_DALIAddressType} [547]).
  - \texttt{nInstanceAddress}: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
  - \texttt{eInstanceAddressType}: Defines the access type to the required instance within the DALI control device (see \texttt{E_DALIInstanceAddressType} [554]).
  - \texttt{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \texttt{E_DALICommandPriority} [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

- **VAR_OUTPUT**
  - \texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.
  - \texttt{ipResultMessage}: Interface pointer (see \texttt{Error evaluation} [574]) that can be used to obtain detailed information about the processing of the function block (see \texttt{Runtime messages} [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.
  - \texttt{bBusy}: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.
nReport: Value of the variable \textit{tReport} [1 s]. The output value is between 0 and 255, as a multiple of the step size of 1 s.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.12.6 \textbf{FB\_DALI304\_SetDeadtimeTimer}

The function block stores the value of \textit{nDeadtime} in the variable \textit{tDeadtime} [50 ms] of the DALI control device.

The value is written in steps of 50 ms. The permissible value range is 0 s to 12.75 s. It is entered in numbers from 0 to 255.

The function block cannot be used when using the KL6811.

\textbf{VAR\_INPUT}

\begin{verbatim}
  bStart          : BOOL;
nAddress         : BYTE;
eAddressType     : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nDeadtime        : Byte := 30;
\end{verbatim}

\textbf{bStart}: Execution of the DALI commands is triggered via a positive edge at this input.

\textbf{nAddress}: Address of a DALI device or a DALI group.

\textbf{eAddressType}: Defines whether the input \textit{nAddress} contains a short address (0...63) or a group address (0...31). Input \textit{nAddress} has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see \textit{E\_DALIAddressType} [547]).

\textbf{nInstanceAddress}: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

\textbf{eInstanceAddressType}: Defines the access type to the required instance within the DALI control device (see \textit{E\_DALIInstanceAddressType} [554]).

\textbf{eCommandPriority}: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \textit{E\_DALICommandPriority} [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

\textbf{nDeadtime}: Value of the variable \textit{tDeadtime} [50 ms]. The input value is between 0 and 255, as a multiple of the step size of 50 ms.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.1.2.12.7 FB_DALI304SetHysteresis

The function block stores the value nHysteresis in the variable hysteresis [341] of the DALI control device. The value is stated in %. The permissible value range is between 0% and 25%.

The function block cannot be used when using the KL6811.

VAR_INPUT

bStart : BOOL;
nAddress : BYTE;
eAddressType : E_DALIAddressType := E_DALIAddressType.Short;
nInstanceAddress : BYTE := 0;
eInstanceAddressType : E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
eCommandPriority : E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
nHysteresis : BYTE := 5;

bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0...63) or a group address (0...31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nInstanceAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).
**eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see \E_DALICommandPriority\.[p 549]).

The command priorities are supported by the KL6821 from the firmware version BD.

**nHysteresis**: Input value of the variable *hysteresis* in %.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation.[p 574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages.[p 561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### 3.1.2.12.8 **FB_DALI304SetHysteresisMin**

The function block stores the value `nHysteresisMin` in the variable `hysteresisMin`.[p 340] of the DALI control device.

Since the hysteresis is given as a percentage, the absolute value of the hysteresis depends on the actual measured value. For very small measured values, the hysteresis is therefore also very small. This causes unnecessary events to be sent. For this reason, a minimum possible hysteresis can be set.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

- **bStart**: BOOL;
- **nAddress**: BYTE;
- **eAddressType**: E_DALIAddressType := E_DALIAddressType.Short;
- **nInstanceAddress**: BYTE := 0;
- **eInstanceAddressType**: E_DALIInstanceAddressType := E_DALIInstanceAddressType.InstanceNumber;
- **eCommandPriority**: E_DALICommandPriority := E_DALICommandPriority.MiddleLow;
- **nHysteresisMin**: Byte := 10;
bStart: Execution of the DALI commands is triggered via a positive edge at this input.

nAddress: Address of a DALI device or a DALI group.

eAddressType: Defines whether the input nAddress contains a short address (0…63) or a group address (0…31). Input nAddress has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).

nINSTANCEAddress: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.

eInstanceAddressType: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).

eCommandPriority: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).

The command priorities are supported by the KL6821 from the firmware version BD.

nHysteresisMin: Value of the smallest possible hysteresis hysteresisMin.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>bError</th>
<th>BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

**3.1.2.12.9 FB_DALI304SetReportTimer**

The function block stores the value of nReport in the variable tReport [340] of the DALI control device.

The value is written in steps of 1 s. The permissible value range is 0 s to 4 min 15 s. It is entered in numbers from 0 to 255.
The function block cannot be used when using the KL6811.

### VAR_INPUT

- **bStart**: Execution of the DALI commands is triggered via a positive edge at this input.
- **nAddress**: Address of a DALI device or a DALI group.
- **eAddressType**: Defines whether the input *nAddress* contains a short address (0…63) or a group address (0…31). Input *nAddress* has no meaning if a broadcast or a broadcast to unaddressed devices (BroadcastUnaddr) has been selected (see E_DALIAddressType [547]).
- **nInstanceAddress**: Instance number within a DALI control device. A DALI control device can support multiple instance numbers.
- **eInstanceAddressType**: Defines the access type to the required instance within the DALI control device (see E_DALIInstanceAddressType [554]).
- **eCommandPriority**: Priority (low, middle low, middle, middle high, high) with which the DALI commands are sent (see E_DALICommandPriority [549]).
- **nReport**: Value of the variable *tReport* [1 s]. The input value is between 0 and 255, as a multiple of the step size of 1 s.

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.
- **bBusy**: The output is set as soon as execution of the DALI commands has commenced. It remains active until all DALI commands have been processed.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
### 3.1.2.12.10 Variables

Each instance of a DALI control device has a certain number of variables (parameters). These variables are used to configure the instance and thus influence its behavior. The values of the variables are stored in the respective DALI control device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Reset value</th>
<th>Scope</th>
<th>Size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventFilter</td>
<td>2#0000_0001</td>
<td>2#0000_000X</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tReport</td>
<td>30</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>tDeadtime</td>
<td>30</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>hysteresisMin</td>
<td>See table</td>
<td>0…255</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>hysteresis</td>
<td>5</td>
<td>0…25</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>

X: not specified

**eventFilter**

Specific events for each instance are activated or deactivated with the event filter (see Introduction [328]). If the respective bit is set, the corresponding event is active.

The function block FB_DALI103QueryEventFilter [127] can be used to read the value of the event filter; the function block FB_DALI103SetEventFilter [114] can be used to write to it.

**tReport**

The variable `tReport` specifies the report time of a brightness sensor after the occurrence of an event. The report time is restarted when a new event is sent. The value is given in steps of 1 s. The permissible value range is 0 (0 s) to 255 (4 min 15 s).

The function block FB_DALI304QueryReportTimer [334] can be used to query the value, the function block FB_DALI304SetReportTimer [338] can be used to specify it.

**tDeadtime**

The variable defines the time value for a dead time in which no new event is sent until the dead time has elapsed. If this timer is active, it is restarted after each transmitted event.

If the value is 0, the dead time event is disabled.

The value is given in steps of 50 ms. The permissible value range is 0 to 255 (12.75 s).

The function block FB_DALI304QueryDeadtimeTimer [330] can be used to query the value, the function block FB_DALI304SetDeadtimeTimer [335] can be used to specify it.

**hysteresisMin**

The variable `hysteresisMin` prevents large fluctuations in the illumination at low illuminance and thus low-percentage hysteresis.

The variable is an absolute value with a value range between 0 and 255.

The reset value depends on the resolution of the input values specified by the vendor.
### 3.1.3 Communication

#### 3.1.3.1 FB_KL6811Communication

The function blocks for the DALI commands do not directly access the process image of the KL6811, but place the individual DALI commands in a command buffer. The function block FB_KL6811Communication sequentially reads the DALI commands from the command buffer and forwards them to the KL6811. This prevents multiple function blocks accessing the KL6811 process image at the same time.

Most function blocks have the input `eCommandPriority`, which has no function when using the KL6811. Priorities for the DALI commands are only supported from DALI-2 onwards.

One instance of the function block FB_KL6811Communication is required per KL6811. This instance must be called in a separate, faster task. This faster communication task must have a higher priority than the task in which the function blocks for the individual DALI commands are called.

---

<table>
<thead>
<tr>
<th>resolution</th>
<th>Vendor-dependent reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>40</td>
</tr>
<tr>
<td>13</td>
<td>81</td>
</tr>
<tr>
<td>14</td>
<td>163</td>
</tr>
<tr>
<td>&gt;15</td>
<td>255</td>
</tr>
</tbody>
</table>
The utilization rate of the command buffer can be determined using the outputs of the function block. If you find that the command buffer is overflowing regularly, you should take the following steps:

- How heavily are the individual PLC tasks utilized? TwinCAT provides suitable analysis tools.
- Try to reduce the cycle time of the task in which the function block FB_KL6811Communication is called. The value should not be greater than 6 ms. The optimum value is 2 ms or less.
- Check the cycle time of the PLC task in which the function blocks for the individual DALI commands are called. This value should not be greater than 10 ms. The optimum value is 8 ms or less.
- If possible avoid polling (regular reading) of values. Only read values when they are actually required.
- Distribute the individual control gears evenly over several DALI lines. Since several DALI lines are processed simultaneously in each PLC cycle, this increases the data throughput.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bResetMaximumDemandCounter</td>
<td>A positive edge resets the stored value for the maximum utilization of the command buffer, nBufferMaximumDemandMeter (0...100%).</td>
</tr>
<tr>
<td>bResetOverflowCounter</td>
<td>A positive edge resets the stored value for the number of overflows of the command buffer, nBufferOverflowCounter.</td>
</tr>
<tr>
<td>nOptions</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bInitialise</td>
<td>Configuration of the Bus Terminal is started by a positive edge at this input. Initialization is also carried out automatically when the controller is started. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>bDisableInternalPowerSupply</td>
<td>Defines the operation mode of the internal DALI power supply. The value is written to the terminal by a positive edge at input bInitialise and stored there persistently.</td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bBusy</td>
<td>The output is set as soon as execution of the DALI commands has started. It remains active until the DALI command has been processed.</td>
</tr>
<tr>
<td>nBufferDemandMeter</td>
<td>Utilization rate of the command buffer (0...100 %).</td>
</tr>
<tr>
<td>nBufferMaximumDemandMeter</td>
<td>Maximum utilization rate of the command buffer reached so far (0...100 %). The counter can be reset via the input bResetMaximumDemandCounter.</td>
</tr>
<tr>
<td>nBufferOverflowCounter</td>
<td>Number of command buffer overflows to date. The counter can be reset via the input bResetOverflowCounter.</td>
</tr>
</tbody>
</table>
**bCollisionErrorFC:** If the output is TRUE, a DALI collision was detected on the forward channel. This can mean that a collision with the send data of another DALI master was detected during the transmission of a DALI telegram. The output is automatically set to FALSE again as soon as no collision has occurred on the forward channel during transmission.

**bCollisionErrorBC:** If the output is TRUE, a DALI collision was detected on the backward channel. This can mean that a collision with the send data of another DALI slave was detected during the transmission of a DALI telegram. The output is automatically set to FALSE again as soon as no collision has occurred on the backward channel during transmission.

**bPowerSupplyError:** If the output is TRUE, an overload was detected when using the internal DALI power supply unit of the Bus Terminal (bus undervoltage).

**blInitialising:** During initialization of the Bus Terminal, the output is set and remains active until initialization has been completed. Initialization is also carried out automatically when the controller is started. During this time no DALI commands are processed.

**nTerminalDescription:** Contains the Terminal Name (e.g. 6811). This corresponds to register 8 of the Bus Terminal.

**nFirmwareVersion:** Contains the firmware version. This corresponds to register 9 of the Bus Terminal.

**VAR_IN_OUT**

```
stInData : ST_KL6811InData;
stOutData : ST_KL6811OutData;
```

**stInData:** Structure in the input process image of the KL6811 (see ST_KL6811InData [546]). It is used for communication from the KL6811 to the PLC.

**stOutData:** Structure in the output process image of the KL6811 (see ST_KL6811OutData [547]). It is used for communication from the PLC to the KL6811.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**3.1.3.2 FB_KL6821Communication**

The function blocks for the DALI commands do not directly access the process image of the KL6821, but place the individual DALI commands in a command buffer. The function block FB_KL6821Communication sequentially reads the DALI commands from the command buffer and forwards them to the KL6821. This prevents multiple function blocks accessing the KL6821 process image at the same time.

The function block also reads the events of the DALI control devices from the KL6821 and stores them in a special table. The function blocks FB_DALIGetInputNotification [543] and FB_DALIGetPowerCycleNotification [545] are used to filter out the desired events from this table.
One instance of the function block FB_KL6821Communication must be created for each KL6821. This instance must be called in a separate, faster task. This faster communication task must have a higher priority than the task in which the function blocks for the individual DALI commands are called.

The utilization rate of the command buffer can be determined using the outputs of the function block. Three parameters are output for this purpose. If you find that the command buffer is overflowing regularly, you should take the following steps:

- How heavily are the individual PLC tasks utilized? TwinCAT provides suitable analysis tools.
- Try to reduce the cycle time of the task in which the function block FB_KL6821Communication is called. The value should not be greater than 6 ms. The optimum value is 2 ms or less.
- Check the cycle time of the PLC task in which the function blocks for the individual DALI commands are called. This value should not be greater than 10 ms. The optimum value is 8 ms or less.
- If possible avoid polling (regular reading) of values. Only read values when they are actually required.
- Distribute the individual control gears evenly over several DALI lines. Since several DALI lines are processed simultaneously in each PLC cycle, this increases the data throughput.

### VAR_INPUT

- **bResetMaximumDemandCounter**: A positive edge resets the stored value for the maximum utilization of the command buffer, \( nBufferMaximumDemandMeter \) (0…100 %).
- **bResetOverflowCounter**: A positive edge resets the stored value for the number of overflows of the command buffer, \( nBufferOverflowCounter \).
- **nOptions**: Reserved for future extensions.
- **bResetInactiveProcessImage**: A positive edge cancels the blocking of the process image of the terminal. The output \( bProcessImageInactive \) is reset to FALSE. The blocking is activated as soon as one of the two digital inputs on the terminal has been actuated and the input parameter \( bDoNotLockProcessImage \) is FALSE.
- **bInitialise**: Configuration of the Bus Terminal is started by a positive edge at this input. Initialization is also carried out automatically when the controller is started. During this time no DALI commands are processed.
- **eCommandKBusWatchdog**: Defines the DALI command that is sent as soon as the Bus Terminal is no longer addressed via the K-bus. The value is written to the terminal by a positive edge at input \( bInitialise \) and stored there persistently.
- **eCommandDI1RisingEdge, eCommandDI2RisingEdge**: Defines the DALI command that is sent as soon as a rising edge is detected at the respective input of the Bus Terminal. The value is written to the terminal by a positive edge at input \( bInitialise \) and stored there persistently.
- **eCommandDI1FallingEdge, eCommandDI2FallingEdge**: Defines the DALI command that is sent as soon as a falling edge is detected at the respective input of the Bus Terminal. The value is written to the terminal by a positive edge at input \( bInitialise \) and stored there persistently.
- **bDoNotLockProcessImage**: Defines whether the process image for the PLC is not blocked by actuating the digital inputs (see also \( bInactiveProcessImage \)). The value is written to the terminal by a positive edge at input \( bInitialise \) and stored there persistently.
- **eDisablePowerSupply**: Defines the operation mode of the internal DALI power supply. The value is written to the terminal by a positive edge at input \( bInitialise \) and stored there persistently.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bBusy : BOOL;
nBufferDemandMeter : BYTE;
nBufferMaximumDemandMeter : BYTE;
nBufferOverflowCounter : UINT;
bDigitalInputnActive : BOOL;
bProcessImageInactive : BOOL;
bCollisionError : BOOL;
bPowerSupplyError : BOOL;
bShortCircuit : BOOL;
bInitialising : BOOL;
nTerminalDescription : WORD;
nFirmwareVersion : WORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bBusy: The output is set as soon as the function block processes a DALI command and remains active until the DALI command has been processed.

nBufferDemandMeter: Utilization rate of the command buffer (0…100 %).

nBufferMaximumDemandMeter: Maximum utilization rate of the command buffer reached so far (0…100 %). The counter can be reset via the input bResetMaximumDemandCounter.

nBufferOverflowCounter: Number of command buffer overflows to date. The counter can be reset via the input bResetOverflowCounter.

bDigitalInput1Active, bDigitalInput2Active: One of the digital inputs at the terminal was actuated or is actuated (see also terminal documentation). If the input bDoNotLockProcessImage is not set, the output bProcessImageInactive is set and no further DALI commands can be sent via the Bus Terminal.

bProcessImageInactive: One of the two digital inputs on the Bus Terminal was actuated and bDoNotLockProcessImage is initialized with FALSE. No further DALI commands can be sent from the PLC via the Bus Terminal. The blockage can be released again via the input bResetInactiveProcessImage.

bCollisionError: A data collision was detected on the DALI bus while a DALI command was sent.

bPowerSupplyError: When using the internal DALI power supply unit: Power supply unit fault detected.

bShortCircuit: The 24 V DC supply voltage at connections 1 and 5 of the KL6821 is missing, or a short circuit has been detected on the DALI bus.

bInitialising: During initialization of the Bus Terminal, the output is set and remains active until initialization has been completed. Initialization is also carried out automatically when the controller is started. During this time no DALI commands are processed.

nTerminalDescription: Contains the Terminal Name (e.g. 6821). This corresponds to register 8 of the Bus Terminal.

nFirmwareVersion: Contains the firmware version. This corresponds to register 9 of the Bus Terminal.

VAR_IN_OUT

stInData : ST_KL6821InData;
stOutData : ST_KL6821OutData;

stInData: Structure in the input process image of the KL6821 (see ST_KL6821InData [547]). It is used for communication from the KL6821 to the PLC.

stOutData: Structure in the output process image of the KL6821 (see ST_KL6821OutData [547]). It is used for communication from the KL6821 to the PLC.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4 Devices

All certified DALI-2 devices are listed in a product database on the homepage of the Digital Illumination Interface Alliance ([https://www.dali-alliance.org/products](https://www.dali-alliance.org/products)).

Each certified DALI-2 device has a unique GTIN (Global Trade Item Number). This GTIN can be read via memory bank 0 within the addresses 0x03 to 0x08 with the help of the function blocks `FB_DALI102ReadMemoryBank[31]` and `FB_DALI103ReadMemoryBank[35]`. The memory bank can also be read out with the PLC HMI (see [PLC HMI](#)) and the KS2000 (see [KS2000](#)).

The devices that are supported by a function block in the library are listed below:

#### B.E.G.

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indoor 180-BMS DALI-2</td>
<td>FB_DALI_BEG_Indoor_180_BMS_V8[352]</td>
<td>4007529935406</td>
</tr>
<tr>
<td>LC-Mini 120-BMS DALI-2</td>
<td>FB_DALI_BEG_LC_Mini_120_BMS_V8[355]</td>
<td>4007529935413</td>
</tr>
<tr>
<td>PD11-BMS-FLAT DALI-2</td>
<td>FB_DALI_BEG_PD11_BMS_V8_Flat[358]</td>
<td>4007529935420</td>
</tr>
<tr>
<td>PD2N-BMS-FC DALI-2</td>
<td>FB_DALI_BEG_PD2N_BMS_V8_FC[361]</td>
<td>4007529935437</td>
</tr>
<tr>
<td>PD4N-BMS DALI-2</td>
<td>FB_DALI_BEG_PD4N_BMS_V8[367]</td>
<td>4007529935468</td>
</tr>
<tr>
<td>PD4-BMS-GH-SM DALI-2</td>
<td>FB_DALI_BEG_PD4_BMS_V8_GH_SM[370]</td>
<td>4007529935451</td>
</tr>
<tr>
<td>PICO-BMS DALI-2</td>
<td>FB_DALI_BEG_PICO_BMS_V8[373]</td>
<td>4007529935475</td>
</tr>
</tbody>
</table>

#### Esylux

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD-C 360/8 BMS DALI-2</td>
<td>FB_DALI_Esylux_PDC_360_BMS[376]</td>
<td>4015120428203</td>
</tr>
<tr>
<td>PD-C 360/24 BMS DALI-2</td>
<td>FB_DALI_Esylux_PDC_360_BMS[376]</td>
<td>4015120428210</td>
</tr>
</tbody>
</table>

#### Glamox

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS DALI-2 PB8 SWITCH PANEL</td>
<td>FB_DALI_Glamox_LMS_PB8_SwitchPanel[380]</td>
<td>7020940423762</td>
</tr>
</tbody>
</table>

#### Loytec

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDALI-BM2</td>
<td>FB_DALI_Loytec_LDALI_BM2[383]</td>
<td>4710901730512</td>
</tr>
<tr>
<td>LDALI-MS4-BT</td>
<td>FB_DALI_Loytec_LDALI_MS4_BT[392]</td>
<td>4710901730741</td>
</tr>
<tr>
<td>LDALI-MS2-BT</td>
<td>FB_DALI_Loytec_LDALI_MS2[387]</td>
<td>4710901730734</td>
</tr>
<tr>
<td>LDALI-MS2</td>
<td>FB_DALI_Loytec_LDALI_MS2[387]</td>
<td>4710901730505</td>
</tr>
</tbody>
</table>
### Lunatone

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DALI-2 CS</td>
<td>FB_DALI Lunatone_C5 [396]</td>
<td>9010342013546</td>
</tr>
<tr>
<td>DALI-2 MC</td>
<td>FB_DALI Lunatone_MC [399]</td>
<td>9010342012730</td>
</tr>
<tr>
<td>DALI-2 MC4L</td>
<td>FB_DALI Lunatone_MC4L [401]</td>
<td>9010342012754</td>
</tr>
<tr>
<td>DALI-2 Cross Switch</td>
<td>FB_DALI Lunatone_Cross_Switch [404]</td>
<td>9010342010224</td>
</tr>
</tbody>
</table>

### Niko

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P46MR for surface mounting</td>
<td>FB_DALI Niko_P46MR [406]</td>
<td>5413736374393</td>
</tr>
<tr>
<td>P46MR for SnapFit flush mounting</td>
<td>FB_DALI Niko_P46MR [406]</td>
<td>5413736374362</td>
</tr>
<tr>
<td>P46MR for flush-mounting box</td>
<td>FB_DALI Niko_P46MR [406]</td>
<td>5413736374348</td>
</tr>
</tbody>
</table>

### Osram

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DALI COUPLER E G2</td>
<td>FB_DALI Osram_Coupler_E_G2 [409]</td>
<td>4062172072212</td>
</tr>
<tr>
<td>DALI COUPLER HF G2</td>
<td>FB_DALI Osram_Coupler_HF_G2 [411]</td>
<td>4062172072199</td>
</tr>
<tr>
<td>DALI COUPLER LS Highbay G2</td>
<td>FB_DALI Osram Coupler_LS_Highbay_G2 [414]</td>
<td>4062172072137</td>
</tr>
<tr>
<td>DALI COUPLER MULTI3 G2</td>
<td>FB_DALI Osram_Coupler_Multi3_G2 [416]</td>
<td>4062172072113</td>
</tr>
<tr>
<td>DALI COUPLER PB G2</td>
<td>FB_DALI Osram Coupler_PushB_G2 [419]</td>
<td>4062172087575</td>
</tr>
<tr>
<td>LS/PD CI G2</td>
<td>FB_DALI Osram_LS_PD_CI_G2 [421]</td>
<td>4062172072069</td>
</tr>
<tr>
<td>LS/PD LI G2</td>
<td>FB_DALI Osram_LS_PD_LI_G2 [423]</td>
<td>4062172072069</td>
</tr>
<tr>
<td>LS/PD LI UF G2</td>
<td>FB_DALI Osram_LS_PD_LI_UF_G2 [426]</td>
<td>4062172072045</td>
</tr>
<tr>
<td>Designation</td>
<td>Function block name</td>
<td>GTIN</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Dual HF</td>
<td>FB_DALI_Steinel_ControlProDualHF [429]</td>
<td>4007841057459</td>
</tr>
<tr>
<td>Dual Tech US</td>
<td>FB_DALI_Steinel_ControlProDualTech [432]</td>
<td>4007841057473</td>
</tr>
<tr>
<td>Dual US</td>
<td>FB_DALI_Steinel_ControlProDualUS [435]</td>
<td>4007841057466</td>
</tr>
<tr>
<td>IR Quattro HD</td>
<td>FB_DALI_Steinel_ControlProIRQuattroHD [438]</td>
<td>4007841057497</td>
</tr>
<tr>
<td>US 360</td>
<td>FB_DALI_Steinel_ControlProUS360 [443]</td>
<td>4007841057534</td>
</tr>
<tr>
<td>Dual Light Sensor AP</td>
<td>FB_DALI_Steinel_DualLightSensor [446]</td>
<td>4007841057411</td>
</tr>
<tr>
<td>Dual Light Sensor AP</td>
<td>FB_DALI_Steinel_DualLightSensor [446]</td>
<td>4007841057442</td>
</tr>
<tr>
<td>Dual Light Sensor AP</td>
<td>FB_DALI_Steinel_DualLightSensor [446]</td>
<td>4007841057428</td>
</tr>
<tr>
<td>Hallway IPD UP</td>
<td>FB_DALI_Steinel_Hallway_IPD [448]</td>
<td>4007841064532</td>
</tr>
<tr>
<td>Hallway IPD AP</td>
<td>FB_DALI_Steinel_Hallway_IPD [448]</td>
<td>4007841064549</td>
</tr>
<tr>
<td>Hallway IPD ECO UP</td>
<td>FB_DALI_Steinel_Hallway_IPD_ECO [453]</td>
<td>4007841079048</td>
</tr>
<tr>
<td>Hallway IPD ECO AP</td>
<td>FB_DALI_Steinel_Hallway_IPD_ECO [453]</td>
<td>4007841079031</td>
</tr>
<tr>
<td>HF 3360 AP</td>
<td>FB_DALI_Steinel_HF3360 [458]</td>
<td>4007841066284</td>
</tr>
<tr>
<td>HF 3360 AP square</td>
<td>FB_DALI_Steinel_HF3360 [458]</td>
<td>4007841057312</td>
</tr>
<tr>
<td>HF 3360 AP round</td>
<td>FB_DALI_Steinel_HF3360 [458]</td>
<td>4007841057329</td>
</tr>
<tr>
<td>HF 3360 UP square</td>
<td>FB_DALI_Steinel_HF3360 [458]</td>
<td>4007841066352</td>
</tr>
<tr>
<td>HF 3360 UP round</td>
<td>FB_DALI_Steinel_HF3360 [458]</td>
<td>4007841057336</td>
</tr>
<tr>
<td>HF 360</td>
<td>FB_DALI_Steinel_HF360 [461]</td>
<td>4007841057480</td>
</tr>
<tr>
<td>HF 360 II IPD UP</td>
<td>FB_DALI_Steinel_HF360 II IPD [464]</td>
<td>4007841064280</td>
</tr>
<tr>
<td>HF 360 II IPD AP</td>
<td>FB_DALI_Steinel_HF360 II IPD [464]</td>
<td>4007841064297</td>
</tr>
<tr>
<td>HF 360 II IPD ECO UP</td>
<td>FB_DALI_Steinel_HF360 II IPD ECO [467]</td>
<td>4007841067465</td>
</tr>
<tr>
<td>HF 360 II IPD ECO AP</td>
<td>FB_DALI_Steinel_HF360 II IPD ECO [467]</td>
<td>4007841067458</td>
</tr>
<tr>
<td>IR Micro</td>
<td>FB_DALI_Steinel_IRMicro [470]</td>
<td>4007841057732</td>
</tr>
<tr>
<td>IR Quattro Micro</td>
<td>FB_DALI_Steinel_IRQuattroMicro [472]</td>
<td>4007841053871</td>
</tr>
<tr>
<td>IR Quattro Slim XS</td>
<td>FB_DALI_Steinel_IRQuattroSlimXS [475]</td>
<td>4007841065034</td>
</tr>
<tr>
<td>IS 345 AP</td>
<td>FB_DALI_Steinel_IS345 [477]</td>
<td>4007841057527</td>
</tr>
<tr>
<td>IS 345 AP square</td>
<td>FB_DALI_Steinel_IS345 [477]</td>
<td>4007841066253</td>
</tr>
<tr>
<td>IS 345 AP round</td>
<td>FB_DALI_Steinel_IS345 [477]</td>
<td>4007841057275</td>
</tr>
<tr>
<td>IS 345 UP</td>
<td>FB_DALI_Steinel_IS345 [477]</td>
<td>4007841057299</td>
</tr>
<tr>
<td>IS 345 UP square</td>
<td>FB_DALI_Steinel_IS345 [477]</td>
<td>4007841066321</td>
</tr>
<tr>
<td>IS 345 UP round</td>
<td>FB_DALI_Steinel_IS345 [477]</td>
<td>4007841057282</td>
</tr>
<tr>
<td>IS 345 MX AP</td>
<td>FB_DALI_Steinel_IS345MX [480]</td>
<td>4007841066260</td>
</tr>
<tr>
<td>IS 345 MX AP square</td>
<td>FB_DALI_Steinel_IS345MX [480]</td>
<td>4007841057398</td>
</tr>
<tr>
<td>Designation</td>
<td>Function block name</td>
<td>GTIN</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>IS 345 MX UP</td>
<td>FB_DALI_Steinel_IS345MX</td>
<td>4007841066338</td>
</tr>
<tr>
<td>IS 3180 AP</td>
<td>FB_DALI_Steinel_IS3180</td>
<td>4007841066277</td>
</tr>
<tr>
<td>IS 3180 AP square shape</td>
<td>FB_DALI_Steinel_IS3180</td>
<td>4007841057350</td>
</tr>
<tr>
<td>IS 3180 UP</td>
<td>FB_DALI_Steinel_IS3180</td>
<td>4007841066345</td>
</tr>
<tr>
<td>IS 3180 UP square shape</td>
<td>FB_DALI_Steinel_IS3180</td>
<td>4007841057367</td>
</tr>
<tr>
<td>IS 3360 AP</td>
<td>FB_DALI_Steinel_IS3360</td>
<td>4007841066239</td>
</tr>
<tr>
<td>IS 3360 AP square shape</td>
<td>FB_DALI_Steinel_IS3360</td>
<td>4007841057237</td>
</tr>
<tr>
<td>IS 3360 AP round shape</td>
<td>FB_DALI_Steinel_IS3360</td>
<td>4007841057251</td>
</tr>
<tr>
<td>IS 3360 UP</td>
<td>FB_DALI_Steinel_IS3360</td>
<td>4007841066307</td>
</tr>
<tr>
<td>IS 3360 UP square shape</td>
<td>FB_DALI_Steinel_IS3360</td>
<td>4007841057244</td>
</tr>
<tr>
<td>IS 3360 UP round shape</td>
<td>FB_DALI_Steinel_IS3360</td>
<td>4007841057268</td>
</tr>
<tr>
<td>IS 3360 MX AP</td>
<td>FB_DALI_Steinel_IS3360MX</td>
<td>4007841066246</td>
</tr>
<tr>
<td>IS 3360 MX AP square shape</td>
<td>FB_DALI_Steinel_IS3360MX</td>
<td>4007841057374</td>
</tr>
<tr>
<td>IS 3360 MX UP</td>
<td>FB_DALI_Steinel_IS3360MX</td>
<td>4007841066314</td>
</tr>
<tr>
<td>IS 3360 MX UP square shape</td>
<td>FB_DALI_Steinel_IS3360MX</td>
<td>4007841057381</td>
</tr>
</tbody>
</table>

**Sunricher**

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PushButtonCoupler</td>
<td>FB_DALI_Sunricher_PushButtonCoupler</td>
<td>06971542121819</td>
</tr>
</tbody>
</table>

**Theben**

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlanoSpot 360</td>
<td>FB_DALI_Theben_PlanoSpot_360</td>
<td>4003468200360</td>
</tr>
<tr>
<td>thePassa P360</td>
<td>FB_DALI_Theben_thePassa_P360</td>
<td>4003468200353</td>
</tr>
<tr>
<td>theRonda P360</td>
<td>FB_DALI_Theben_theRonda_P360</td>
<td>4003468200353</td>
</tr>
<tr>
<td>theRonda S360</td>
<td>FB_DALI_Theben_theRonda_S360</td>
<td>4003468200384</td>
</tr>
<tr>
<td>TA 4 S DALI-2</td>
<td>FB_DALI_Theben_TA_4_S</td>
<td>4003468491584</td>
</tr>
</tbody>
</table>

**Tridonic**

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSensor G3 SRC 30 PIR 5DPI</td>
<td>FB_DALI_Tridonic_MSensorG3_SRC_30_PIR</td>
<td>9006210666693</td>
</tr>
<tr>
<td>MSensor G3 SFI 30 PIR 5DPI</td>
<td>FB_DALI_Tridonic_MSensorG3_SFI_30_PIR</td>
<td>9006210672504</td>
</tr>
<tr>
<td>MSensor G3 SFI 30 PIR 16DPI</td>
<td>FB_DALI_Tridonic_MSensorG3_SFI_30_PIR</td>
<td>9006210672502</td>
</tr>
<tr>
<td>MSensor G3 SFI 30 PIR 10DPI</td>
<td>FB_DALI_Tridonic_MSensorG3_SFI_30_PIR</td>
<td>9006210656205</td>
</tr>
<tr>
<td>DALI XC G3 CWM 30 DA2</td>
<td>FB_DALI_Tridonic_XC_G3_CWM30</td>
<td>9006210708881</td>
</tr>
</tbody>
</table>
Zencontrol

<table>
<thead>
<tr>
<th>Designation</th>
<th>Function block name</th>
<th>GTIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR</td>
<td>FB_DALI_Zencontrol_PIR [523]</td>
<td>9336462199091</td>
</tr>
<tr>
<td>Scenepanel Switch</td>
<td>FB_DALI_Zencontrol_Scenepanel_Switch [526]</td>
<td>6971103530319</td>
</tr>
<tr>
<td>Switch 1 Button</td>
<td>FB_DALI_Zencontrol_Switch_1Button [528]</td>
<td>9336462191897</td>
</tr>
</tbody>
</table>

As an alternative to the device function blocks, individual instances of DALI control units can be addressed via the general function blocks. Each of the general function blocks refers to a specific part of IEC 62386.

<table>
<thead>
<tr>
<th>IEC 62386-XXX</th>
<th>Function</th>
<th>Function block name</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>General properties of the control devices</td>
<td>FB_DALI103ControlDevice [530]</td>
</tr>
<tr>
<td>301</td>
<td>Push buttons</td>
<td>FB_DALI301PushButton [533]</td>
</tr>
<tr>
<td>302</td>
<td>Input devices, absolute encoder</td>
<td>FB_DALI302AbsoluteInputDevice [535]</td>
</tr>
<tr>
<td>303</td>
<td>Occupancy sensor</td>
<td>FB_DALI303OccupancySensor [538]</td>
</tr>
<tr>
<td>304</td>
<td>Light sensor</td>
<td>FB_DALI304LightSensor [540]</td>
</tr>
</tbody>
</table>

Further function blocks to simplify the application of device function blocks

<table>
<thead>
<tr>
<th>Function block name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_DALI_ToPushButtonState [542]</td>
<td>Determines the static state from the events of a button.</td>
</tr>
</tbody>
</table>

3.1.4.1 Introduction

Device function blocks

Various manufacturers offer devices that combine several DALI parts in one device.

For easy integration into projects, the Tc3_DALI library contains a constantly growing number of function blocks for certified DALI-2 devices. Additional information on the certification of DALI-2 devices can be found on the homepage of the DALI Alliance (www.dali-alliance.org).

Individual manufacturers frequently offer further information on the devices that describe each parameter in more detail.

All parameters of a DALI control device are mapped via properties on the device function block and can be written to the DALI control device.

The properties have internally predefined default values. Therefore, it is not absolutely necessary to change the properties of a device function block.
The function block represents the Indoor 180-BMS DALI-2 DALI sensor from B.E.G..

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables `bQueryOccupancy` and `bQueryBrightness`.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at `bInitialize`.

- **Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.**

- **The function block cannot be used when using the KL6811.**

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bInitialize</code></td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td><code>nShortAddress</code></td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td><code>nOptions</code></td>
<td>DWORD</td>
<td>0</td>
</tr>
<tr>
<td><code>bQueryOccupancy</code></td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td><code>bQueryBrightness</code></td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td><code>bCancelHoldTimerOccupancy</code></td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

- **`bInitialize`:** The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- **`nShortAddress`:** Short address (0…63) of the DALI control device.

- **`nOptions`:** Reserved for future extensions.

- **`bQueryOccupancy`:** The occupancy is queried immediately by a positive edge at this input.

- **`bQueryBrightness`:** The brightness is queried immediately by a positive edge at this input.

- **`bCancelHoldTimerOccupancy`:** A positive edge at this input prematurely terminates the Hold Timer and resets the output `bOccupied`. 
VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy:** The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>40</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the LC-Mini 120-BMS DALI-2 DALI sensor from B.E.G..

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

- bInitialize : BOOL := FALSE;
- nShortAddress : BYTE := 0;
- nOptions : DWORD := 0;
- bQueryOccupancy : BOOL := FALSE;
- bQueryBrightness : BOOL := FALSE;
- bCancelHoldTimerOccupancy : BOOL := FALSE;

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryOccupancy**: The occupancy is queried immediately by a positive edge at this input.

**bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

**bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

**VAR_OUTPUT**

- bError : BOOL;
- ipResultMessage : I_TcMessage;
- bInitializing : BOOL;
- bReadingOccupancy : BOOL;

- bOccupied
**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy:** The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via *bInitialize* are available as properties.
## Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
| bEnableBrightness | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input $bQueryBrightness$.
| bEnableOccupancy | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input $bQueryOccupancy$.
| eEventPriorityBrightness | E_DALIEventPriority | Get, Set | MiddleLow | Setting of the event priority [160].
| nReportTimerBrightness | BYTE | Get, Set | 30 s | Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.
| nDeadtimeTimerBrightness | UINT | Get, Set | 1500 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
| nHysteresisBrightness | BYTE | Get, Set | 5 % | Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.
| nHysteresisMinBrightness | BYTE | Get, Set | 40 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.
| eEventPriorityOccupancy | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
| nDeadtimeTimerOccupancy | UINT | Get, Set | 100 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
| nHoldTimerOccupancy | UINT | Get, Set | 900 s | Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.
| nReportTimerOccupancy | BYTE | Get, Set | 60 s | Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3.DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the PD11-BMS-FLAT DALI-2 DALI sensor from the company B.E.G.

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables \texttt{bQueryOccupancy} and \texttt{bQueryBrightness}.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at \texttt{bInitialize}.

---

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

---

The function block cannot be used when using the KL6811.

---

**VAR_INPUT**

- \texttt{bInitialize} \hspace{1cm} : \texttt{BOOL} \hspace{0.5cm} := \texttt{FALSE};
- \texttt{nShortAddress} \hspace{1cm} : \texttt{BYTE} \hspace{0.5cm} := \texttt{0};
- \texttt{nOptions} \hspace{1cm} : \texttt{DWORD} \hspace{0.5cm} := \texttt{0};
- \texttt{bQueryOccupancy} \hspace{1cm} : \texttt{BOOL} \hspace{0.5cm} := \texttt{FALSE};
- \texttt{bQueryBrightness} \hspace{1cm} : \texttt{BOOL} \hspace{0.5cm} := \texttt{FALSE};
- \texttt{bCancelHoldTimerOccupancy} \hspace{1cm} : \texttt{BOOL} \hspace{0.5cm} := \texttt{FALSE};

\texttt{bInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\texttt{nShortAddress}: Short address (0…63) of the DALI control device.

\texttt{nOptions}: Reserved for future extensions.

\texttt{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

\texttt{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

\texttt{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \texttt{bOccupied}.

**VAR_OUTPUT**

- \texttt{bError} \hspace{1cm} : \texttt{BOOL};
- \texttt{ipResultMessage} \hspace{1cm} : \texttt{I_TcMessage};
- \texttt{bInitializing} \hspace{1cm} : \texttt{BOOL};
- \texttt{bReadingOccupancy} \hspace{1cm} : \texttt{BOOL};
**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via *bInitialize* are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBright</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>40</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the PD2N-BMS-FC DALI-2 DALI sensor from B.E.G..

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>False</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>0</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>False</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>False</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>False</td>
</tr>
</tbody>
</table>

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryOccupancy**: The occupancy is queried immediately by a positive edge at this input.

**bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

**bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

bError: The function block represents the PD2N-BMS-FC DALI-2 DALI sensor from B.E.G.
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [p 574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [p 561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>40</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the PD2N-BMS-V8-FM DALI-2 DALI sensor from B.E.G..

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>
**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy:** The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>40</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the PD4N-BMS DALI-2 DALI sensor from B.E.G..

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

---

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.</td>
</tr>
</tbody>
</table>

---

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBright</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>Brightness</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBright</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>40</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>Brightness</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the PD4-BMS-GH-SM DALI-2 DALI sensor from B.E.G..

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

- **VAR_INPUT**
  - bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
  - nShortAddress: Short address (0…63) of the DALI control device.
  - nOptions: Reserved for future extensions.
  - bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.
  - bQueryBrightness: The brightness is queried immediately by a positive edge at this input.
  - bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

- **VAR_OUTPUT**
  - bError:
  - ipResultMessage:
  - bInitializing:
  - bReadingOccupancy:

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.
PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
bEnableBrightness | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.
bEnableOccupancy | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.
eEventPriorityBrightness | E_DALIEventPriority | Get, Set | MiddleLow | Setting of the event priority [160].
nReportTimerBrightness | BYTE | Get, Set | 30 s | Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.
nDeadt imeTimerBrightness | UINT | Get, Set | 1500 ms | Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
nHysteresisBrightness | BYTE | Get, Set | 5 % | Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.
nHysteresisMinBrightness | BYTE | Get, Set | 40 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.
eEventPriorityOccupancy | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
nDeadt imeTimerOccupancy | UINT | Get, Set | 100 ms | Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
nHoldTimerOccupancy | UINT | Get, Set | 900 s | Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.
nReportTimerOccupancy | BYTE | Get, Set | 60 s | Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the PICO-BMS DALI-2 DALI sensor from B.E.G.. This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

- bInitialize : BOOL := FALSE;
- nShortAddress : BYTE := 0;
- nOptions : DWORD := 0;
- bQueryOccupancy : BOOL := FALSE;
- bQueryBrightness : BOOL := FALSE;
- bCancelHoldTimerOccupancy : BOOL := FALSE;

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryOccupancy**: The occupancy is queried immediately by a positive edge at this input.

**bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

**bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

**VAR_OUTPUT**

- bError : BOOL;
- ipResultMessage : I_TcMessage;
- bInitializing : BOOL;
- bReadingOccupancy : BOOL;

- bOccupied
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority.[160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>40</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.[160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
3.1.4.3 Esylux

3.1.4.3.1 FB_DALI_Esylux_PDC_360_BMS

The function block represents the PD-C 360/8 BMS DALI-2 or PD-C 360/24 BMS DALI-2 DALI sensor from Esylux.

This outputs two signals from connected push buttons (instances 0 and 1), an occupancy signal (instance 2) and the measured brightness value (instance 3).

Further information on the supported instance types can be found here:

Part 301 (push buttons) – Introduction [P. 288]

Part 303 (occupancy sensors) – Introduction [P. 313]

Part 304 (light sensors) – Introduction [P. 327]

The existing parameters can be overwritten by changing the properties listed below.

The input variables bQueryPushButtonN, bQueryOccupancy and bQueryBrightness can be used to trigger the read-out immediately, even if the corresponding instance is not enabled.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

ALS correction factor

During programming, it may be necessary to calibrate the brightness measurement.

The values can be changed via the properties.

The function block transmits the individual ALS (ambient light sensor) parameters to the 2 light sensors integrated in the DALI control device.

The ALS correction factor has a value range from 0.1 to 5.0. The input values are increased by a factor of 10 from 1 to 50.

The correction factor of a light sensor is determined at 100 and 500 lux on the floor. A separate luxmeter is necessary for this.

1. Dim the lighting until 100 lux is measured on the floor.
2. Read the output value of the sensor.
3. Determine the correction factor for the sensor and enter it at the inputs \texttt{nALS0CorrectionFactor100} and \texttt{nALS1CorrectionFactor100}.

In the following example, a value of 100 lux is measured with the luxmeter. The device function block returns the value 54 at the output \texttt{nBrightnessLevel}. This calculates the correction factor as follows:

\[
\frac{\text{nALS0CorrectionFactor100}}{\text{nALS1CorrectionFactor100}} = \frac{100}{54} = 1.851 \approx 1.9 \rightarrow 19
\]

The value 19 must be entered at the inputs \texttt{nALS0CorrectionFactor100} and \texttt{nALS1CorrectionFactor100}.

4. Repeat this procedure from step 1 at 500 lux.

VAR\_INPUT
\begin{verbatim}
  bInitialize : BOOL := FALSE;
  nShortAddress : BYTE := 0;
  nOptions : DWORD := 0;
  bQueryPushButton1 : BOOL := FALSE;
  bQueryPushButton2 : BOOL := FALSE;
  bQueryOccupancy : BOOL := FALSE;
  bQueryBrightness : BOOL := FALSE;
  bCancelHoldTimerOccupancy : BOOL := FALSE;
\end{verbatim}

\texttt{blninitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\texttt{nShortAddress}: Short address (0…63) of the DALI control device.

\texttt{nOptions}: Reserved for future extensions.

\texttt{bQueryPushbuttonN}: The state of the button is queried immediately by a positive edge at this input.

\texttt{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

\texttt{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

\texttt{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \texttt{bOccupied}.

VAR\_OUTPUT
\begin{verbatim}
  bError : BOOL;
  ipResultMessage : I_TcMessage;
  bInitializing : BOOL;
  bReadingPushButton1 : BOOL;
  bReadingPushButton2 : BOOL;
  bReadingOccupancy : BOOL;
  bReadingBrightness : BOOL;
  nPushButton1Event : WORD;
  nPushButton2Event : WORD;
  bOccupied : BOOL;
  nBrightnessLevel : UINT;
\end{verbatim}

\texttt{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\texttt{ipResultMessage}: Interface pointer (see \textit{error evaluation} [574]) that can be used to obtain detailed information about the processing of the function block (see \textit{runtime messages} [561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\texttt{blninitialize}: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

\texttt{bReadingPushButton1}: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.
**bReadingPushButton2**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bReadingOccupancy**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**nPPushButtonNEvent**: The output outputs the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block **FB_DALI_ToPushButtonState** can be used to convert the state of a push button into a BOOL variable.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device. A description can be found under **ALS correction factor**.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input 1 or 2, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle Low</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nALS0CorrectionFactor100</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>Correction factor (1…50) of light sensor 0 at a measured value of 100 lux on the surface to be measured.</td>
</tr>
<tr>
<td>nALS0CorrectionFactor500</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>Correction factor (1…50) of light sensor 0 at a measured value of 500 lux on the surface to be measured.</td>
</tr>
<tr>
<td>nALS1CorrectionFactor100</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>Correction factor (1…50) of light sensor 1 at a measured value of 100 lux on the surface to be measured.</td>
</tr>
<tr>
<td>nALS1CorrectionFactor500</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>Correction factor (1…50) of light sensor 1 at a measured value of 500 lux on the surface to be measured.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimeBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>81</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimeOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>95</td>
<td>Setting of the sensitivity of both occupancy sensor instances from 1 (low sensitivity) to 100 (high sensitivity). The written value of 254 resets the sensitivity to the factory setting.</td>
</tr>
</tbody>
</table>
**Programming**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**3.1.4.4 Glamox**

**3.1.4.4.1 FB_DALI_Glamox_LMS_P8_SwitchPanel**

The function block represents the LMS DALI-2 PB8 Switch Panel DALI push-button interface from Glamox.

Up to eight push buttons (instances 0 to 7) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]
The existing parameters can be overwritten by changing the properties listed below.

Readout of the corresponding instance can be triggered immediately via the input variables `bQueryInputValueN`.

The properties `bEnablePushButtonN` make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at `bInitialize` and stored there.

---

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>0</td>
</tr>
<tr>
<td>bQueryInputValue1</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue2</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue3</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue4</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue5</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue6</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue7</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bQueryInputValue8</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryInputValueN**: The state of the corresponding push button is queried immediately by a positive edge at this input.

---

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue1</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue2</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue3</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue4</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue5</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue6</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue7</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue8</td>
<td>BOOL</td>
</tr>
<tr>
<td>nPushButton1Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton2Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton3Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton4Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton5Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton6Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton7Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton8Event</td>
<td>WORD</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.
bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

**PROPERTIES**

All parameters that are written to the DALI control device via binitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100…2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>
3.1.4.5 Loytec

3.1.4.5.1 FB_DALI_Loytec_LDALI_BM2

The function block represents the LDALI-BM2 DALI push-button interface from Loytec.

Up to four push buttons (instances 0, 3, 6, and 7) can be connected directly via the device.

The first two inputs are configurable so that NTC/PTC temperature sensors (instances 1 and 4) or potentiometers/slide resistors (instances 2 and 5) can be used instead of push buttons.

These functions can be configured via the properties `eInput1Configuration` and `eInput2Configuration` (see `E_DALILoytecInputConfiguration`[555]).

The resistance value is read directly on setting one of the two inputs as Generic Input (resistance thermometer). This readout takes place when a positive edge is detected at the respective input `bQueryInputValue1` or `bQueryInputValue2`.

The resistance value is read in % when setting one of the two inputs as Absolute Input (potentiometer). The corresponding channel is configured via the property `eResistanceValueAbsoluteInput1` or `eResistanceValueAbsoluteInput2` on the connected resistor (1 kΩ or 10 kΩ) (see Use of properties). The output value lies between 0 (0 %) and 31 (100 %).

The inputs can be queried via a positive edge at `bQueryInputValueN`.

In addition, the time-dependent querying of these values is possible. The properties `nReportTimerAbsoluteInput1` and `nReportTimerAbsoluteInput2` are used for this. The value is read again on expiry of the set time [s]. To prevent events following one another too quickly, a dead time [ms] can be set via the properties `nDeadtimeTimerAbsoluteInput1` and `nDeadtimeTimerAbsoluteInput2`.

The push button inputs 3 and 4 can be disabled via the properties `bEnableInput3` and `bEnableInput4`.

All parameters are written to the DALI control device by a positive edge at `bInitialize` and stored there.

The existing parameters can be overwritten by changing the properties listed below.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- Set the event priorities on the properties `eEventPriorityInput1` and `eEventPriorityInput2` to the value `MiddleLow` if the inputs are not used as push buttons.
The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryInputValue1 : BOOL := FALSE;
bQueryInputValue2 : BOOL := FALSE;
bQueryInputValue3 : BOOL := FALSE;
bQueryInputValue4 : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryInputValueN: The state of the corresponding input is queried immediately by a positive edge at this input. Values of resistance sensors (setting of the input as Generic Input) can only be queried manually and not event-driven.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingInputValue1 : BOOL;
bReadingInputValue2 : BOOL;
bReadingInputValue3 : BOOL;
bReadingInputValue4 : BOOL;
nPushButton1Event : WORD;
nGenericInput1 : UINT;
nAbsoluteInput1 : BYTE;
nPushButton2Event : WORD;
nGenericInput2 : UINT;
nAbsoluteInput2 : BYTE;
nPushButton3Event : WORD;
nPushButton4Event : WORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block `FB_DALI_ToPushButtonState` can be used to convert the state of a push button into a BOOL variable.

**nGenericInputN:** Outputs containing the measured values of the connected resistance sensors (NTC/PTC).

**nAbsoluteInputN:** Outputs containing the measured values of the connected potentiometers or slide resistors. The individual ranges are scaled from 0 to 100 %, with the following output values: 0 = 0 % and 31 = 100 %.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eInputNConfiguration</td>
<td>E_DALILoyteInputConfiguration</td>
<td>Get, Set</td>
<td>PushButton</td>
<td>Configuration of input 1 or 2 as a push button or analog input for the connection of potentiometers or resistance sensors, e.g. NTC/PTC.</td>
</tr>
<tr>
<td>eResistanceValueAbsoluteInputN</td>
<td>E_DALILoyteResistanceConfiguration</td>
<td>Get, Set</td>
<td>Resistanced10kOhm</td>
<td>Configuration of the resistance value of the potentiometer or slide resistor at input 1 or 2.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>Push button input 3 or 4, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>eEventPriorityInputN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerAbsoluteInputN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nReportTimerAbsoluteInputN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>0 s</td>
<td>Time [0…255 s] after which the pending event of the absolute encoder is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the LDALI-MS2-BT or LDALI-MS2 DALI sensor from Loytec.

This outputs a brightness value (instance 0), an occupancy signal (instance 1), a temperature value (instance 2), a relative humidity value (instance 3), three hardware push button messages (instances 4 to 6) and 18 infrared remote control push button messages (instances 7 to 24).

The infrared push button instances can be used via a remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) – Introduction [288]

Part 303 (occupancy sensors) - Introduction [313]

Part 304 (light sensors) - Introduction [327]

The temperature measurement and humidity measurement are executed as Generic Inputs. These values can only be queried via a positive edge at the inputs bQueryTemperature and bQueryHumidity.

For the other instances, the readout can also be triggered manually via the inputs bQueryBrightness and bQueryOccupancy.
The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at \textit{bInitialize}.

- **i** Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- **i** The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>:= FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>:= 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>:= 0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>:= FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>:= FALSE;</td>
</tr>
<tr>
<td>bQueryTemperature</td>
<td>BOOL</td>
<td>:= FALSE;</td>
</tr>
<tr>
<td>bQueryHumidity</td>
<td>BOOL</td>
<td>:= FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>:= FALSE;</td>
</tr>
</tbody>
</table>

\textit{bInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\textit{nShortAddress}: Short address (0…63) of the DALI control device.

\textit{nOptions}: Reserved for future extensions.

\textit{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

\textit{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

\textit{bQueryTemperature}: The temperature is queried immediately by a positive edge at this input. The DALI device does not support transmission of the temperature per event.

\textit{bQueryHumidity}: The relative humidity is queried immediately by a positive edge at this input. The DALI device does not support transmission of relative humidity per event.

\textit{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \textit{bOccupied}.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingTemperature</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingHumidity</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
<tr>
<td>fTemperature</td>
<td>LREAL;</td>
</tr>
<tr>
<td>nHumidity</td>
<td>BYTE;</td>
</tr>
<tr>
<td>nPushButtonHWNEvent</td>
<td>WORD;</td>
</tr>
<tr>
<td>nPushButtonIRNEvent</td>
<td>WORD;</td>
</tr>
</tbody>
</table>

\textit{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.

\textit{ipResultMessage}: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.

\textit{bInitializing}: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.
bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bReadingTemperature: The output is TRUE as long as values of the instance of the temperature sensor are being read by the DALI control device.

bReadingHumidity: The output is TRUE as long as values of the humidity sensor instance are read by the DALI controller.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

fTemperature: Measured temperature value (-5.0...60.0 °C) of the temperature sensor.

nHumidity: Measured value of the relative humidity (0...100 %).

nPUSHButtonHWNEvent: The outputs (1...3) output the events for the respective push button via one bit each for a cycle.

nPUSHButtonIRNEvent: The outputs (1...18) output the events for the corresponding infrared button via one bit for one cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>bEnableTemperature</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for temperature measurement. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input bQueryTemperature.</td>
</tr>
<tr>
<td>bEnableHumidity</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>A TRUE on this property enables the instance to measure relative humidity. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input bQueryHumidity.</td>
</tr>
<tr>
<td>bEnablePushButtonHWN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input 1 to 3, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>bEnablePushButtonIRN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Infrared button inputs 1 to 18, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>255</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
</tbody>
</table>
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
nReportTimerOccupancy | BYTE | Get, Set | 60 s | Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.
eEventPriorityPushButtonHWN | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
nEventFilterPushButtonHWN | BYTE | Get, Set | 2#1000_0011 | Setting of the event filter [288].
nDoubleTimerPushButtonHWN | UINT | Get, Set | 0 s | Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.
nRepeatTimerPushButtonHWN | UINT | Get, Set | 160 ms | Interval (100...2000 ms) of repetitive events in the case of a long keystroke.
nShortTimerPushButtonHWN | UINT | Get, Set | 500 ms | Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.
nStuckTimerPushButtonHWN | BYTE | Get, Set | 20 s | Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.
eEventPriorityPushButtonIRN | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
nEventFilterPushButtonIRN | BYTE | Get, Set | 2#1000_0011 | Setting of the event filter [288].
nDoubleTimerPushButtonIRN | UINT | Get, Set | 0 s | Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.
nRepeatTimerPushButtonIRN | UINT | Get, Set | 160 ms | Interval (100...2000 ms) of repetitive events in the case of a long keystroke.
nShortTimerPushButtonIRN | UINT | Get, Set | 500 ms | Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.
nStuckTimerPushButtonIRN | BYTE | Get, Set | 20 s | Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>
The function block represents the LDALI-MS4-BT DALI sensor from Loytec.

This outputs a brightness value (instance 0), an occupancy signal (instance 1), a temperature value (instance 2), a relative humidity value (instance 3), three hardware push button messages (instances 4 to 6) and 18 infrared remote control push button messages (instances 7 to 24).

The infrared push button instances can be used via a remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer’s device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) – Introduction [288]
Part 303 (occupancy sensors) - Introduction [313]
Part 304 (light sensors) - Introduction [327]

The temperature measurement and humidity measurement are executed as Generic Inputs. These values can only be queried via a positive edge at the inputs bQueryTemperature and bQueryHumidity.

For the other instances, the readout can also be triggered manually via the inputs bQueryBrightness and bQueryOccupancy.
The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at \textit{bInit}. 

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

\textbf{VAR\_INPUT}

\begin{verbatim}
VAR\_INPUT
bInit : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bQueryTemperature : BOOL := FALSE;
bQueryHumidity : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;
\end{verbatim}

\textbf{bInit}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\textbf{nShortAddress}: Short address (0…63) of the DALI control device.

\textbf{nOptions}: Reserved for future extensions.

\textbf{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

\textbf{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

\textbf{bQueryTemperature}: The temperature is queried immediately by a positive edge at this input. The DALI device does not support transmission of the temperature per event.

\textbf{bQueryHumidity}: The relative humidity is queried immediately by a positive edge at this input. The DALI device does not support transmission of relative humidity per event.

\textbf{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \textit{bOccupied}.

\textbf{VAR\_OUTPUT}

\begin{verbatim}
VAR\_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bReadingTemperature : BOOL;
bReadingHumidity : BOOL;
bOccupied : BOOL;
brightnessLevel : UINT;
fTemperature : LREAL;
nHumidity : BYTE;
nPushButtonHWEvent : WORD;
nPushButtonIREvent : WORD;
\end{verbatim}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.

\textbf{ipResultMessage}: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.

\textbf{bInitializing}: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.
**bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bReadingTemperature**: The output is TRUE as long as values of the instance of the temperature sensor are being read by the DALI control device.

**bReadingHumidity**: The output is TRUE as long as values of the humidity sensor instance are read by the DALI controller.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**fTemperature**: Measured temperature value (-5.0...60.0 °C) of the temperature sensor.

**nHumidity**: Measured value of the relative humidity (0...100 %).

**nPUSHBUTTONHWNEVENT**: The outputs (1…3) output the events for the respective push button via one bit each for a cycle.

**nPUSHBUTTONIRNEVENT**: The outputs (1…18) output the events for the corresponding infrared button via one bit for one cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block `FB_DALI_ToPushButtonState` can be used to convert the state of a push button into a BOOL variable.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>bEnableTemperature</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for temperature measurement. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input bQueryTemperature.</td>
</tr>
<tr>
<td>bEnableHumidity</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>A TRUE on this property enables the instance to measure relative humidity. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input bQueryHumidity.</td>
</tr>
<tr>
<td>bEnablePushButtonHW</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input 1 to 3, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>bEnablePushButtonIR</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Infrared button inputs 1 to 18, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBright</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>255</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
</tbody>
</table>
### Name and Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>eEventPriorityPushButtonHWN</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td></td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nEventFilterPushButtonHWN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonHWN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 s</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonHWN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonHWN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonHWN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
<tr>
<td>eEventPriorityPushButtonIRN</td>
<td>E_DALIE</td>
<td>Get, Set</td>
<td></td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nEventFilterPushButtonIRN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonIRN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 s</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonIRN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonIRN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonIRN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.424.13</td>
<td>Tc3_DALI from v3.6.0.0</td>
</tr>
</tbody>
</table>

3.1.4.6 Lunatone

3.1.4.6.1 FB_DALI_Lunatone_CS

The function block represents the CS DALI-2 sensor from Lunatone.
Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

Readout can be triggered manually via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bInitializing</td>
<td>The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.</td>
</tr>
</tbody>
</table>
**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

---

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.9.0.0</td>
</tr>
</tbody>
</table>

3.1.4.6.2  **FB_DALI_Lunatone_MC**

The function block represents the MC DALI push-button interface from Lunatone.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:
Part 301 (push buttons) - [Introduction](#)

The existing parameters can be overwritten by changing the properties listed below.

The input variables `bQueryInputValueN` trigger the manual reading of the corresponding instance.

The properties `bEnablePushButtonN` make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at `bInitialize` and stored there.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryInputValue1</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryInputValue2</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryInputValue3</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryInputValue4</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

`bInitialize`: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

`nShortAddress`: Short address (0…63) of the DALI control device.

`nOptions`: Reserved for future extensions.
bQueryInputValueN: The state of the corresponding push button is queried immediately by a positive edge at this input.

VAR OUTPUT

<table>
<thead>
<tr>
<th>bError</th>
<th>: BOOL;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipResultMessage</td>
<td>: I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue1</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue2</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue3</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue4</td>
<td>: BOOL;</td>
</tr>
<tr>
<td>nPushButton1Event</td>
<td>: WORD;</td>
</tr>
<tr>
<td>nPushButton2Event</td>
<td>: WORD;</td>
</tr>
<tr>
<td>nPushButton3Event</td>
<td>: WORD;</td>
</tr>
<tr>
<td>nPushButton4Event</td>
<td>: WORD;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Requirements

**Development environment**

<table>
<thead>
<tr>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
</tr>
<tr>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.6.3 FB_DALI_Lunatone_MC4L

The function block represents the MC4L DALI push-button interface from Lunatone.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]

The existing parameters can be overwritten by changing the properties listed below.

The input variables `bQueryInputValueN` trigger the manual reading of the corresponding instance.

The properties `bEnablePushButtonN` make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at `bInitialize` and stored there.
Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryInputValue1</td>
<td>BOOL</td>
<td>The state of the corresponding push button is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryInputValue2</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bQueryInputValue3</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bQueryInputValue4</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
<td>The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.</td>
</tr>
<tr>
<td>bReadingInputValue1</td>
<td>BOOL</td>
<td>The outputs are set as soon as the manual reading of the corresponding input has been triggered.</td>
</tr>
<tr>
<td>bReadingInputValue2</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bReadingInputValue3</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bReadingInputValue4</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>nPushButton1Event</td>
<td>WORD</td>
<td>The outputs output the events for the respective push button via one bit each for a cycle.</td>
</tr>
<tr>
<td>nPushButton2Event</td>
<td>WORD</td>
<td></td>
</tr>
<tr>
<td>nPushButton3Event</td>
<td>WORD</td>
<td></td>
</tr>
<tr>
<td>nPushButton4Event</td>
<td>WORD</td>
<td></td>
</tr>
</tbody>
</table>

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryInputValueN: The state of the corresponding push button is queried immediately by a positive edge at this input.

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState can be used to convert the state of a push button into a BOOL variable.

**PROPERTIES**

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>Push button input 1...4, which can be either enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#10000010</td>
<td>Setting of the event filter.</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3.DALI from v3.9.0.0</td>
</tr>
</tbody>
</table>
3.1.4.6.4  FB_DALI_Lunatone_Cross_Switch

The function block represents the Cross Switch DALI push-button interface from Lunatone.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:
Part 301 (push buttons) - Introduction [288]

The existing parameters can be overwritten by changing the properties listed below.

The input variables \texttt{bQueryInputValueN} trigger the manual reading of the corresponding instance.

The properties \texttt{bEnablePushButtonN} make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at \texttt{bInitialize} and stored there.

\begin{itemize}
\item Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.
\item The function block cannot be used when using the KL6811.
\end{itemize}

\section*{VAR_INPUT}

\begin{itemize}
\item \texttt{bInitialize} : BOOL := FALSE;
\item \texttt{nShortAddress} : BYTE := 0;
\item \texttt{nOptions} : DWORD := 0;
\item \texttt{bQueryInputValue1} : BOOL := FALSE;
\item \texttt{bQueryInputValue2} : BOOL := FALSE;
\item \texttt{bQueryInputValue3} : BOOL := FALSE;
\item \texttt{bQueryInputValue4} : BOOL := FALSE;
\end{itemize}

\texttt{bInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\texttt{nShortAddress}: Short address (0…63) of the DALI control device.

\texttt{nOptions}: Reserved for future extensions.

\texttt{bQueryInputValueN}: The state of the corresponding push button is queried immediately by a positive edge at this input.

\section*{VAR_OUTPUT}

\begin{itemize}
\item \texttt{bError} : BOOL;
\item \texttt{ipResultMessage} : I_TcMessage;
\item \texttt{bInitializing} : BOOL;
\end{itemize}
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>Push button input 1...4, which can be either enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#100_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

### Requirements

**Development environment**

<table>
<thead>
<tr>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
</tr>
<tr>
<td>Tc3_DALI from v3.9.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.7 Niko

#### 3.1.4.7.1 FB_DALI_Niko_P46MR

The function block represents the DALI sensors P46MR for surface mounting, P46MR for SnapFit flush mounting and P46MR for flush-mounting box from Niko.

This outputs an occupancy signal (instance 0) and a brightness value (instance 1).

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

- Part 303 (occupancy sensors) – [Introduction](#) [313]
- Part 304 (light sensors) – [Introduction](#) [327]

Readout can be triggered manually via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.
Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

### VAR_INPUT

- **bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
- **nShortAddress**: Short address (0…63) of the DALI control device.
- **nOptions**: Reserved for future extensions.
- **bQueryOccupancy**: The occupancy is queried immediately by a positive edge at this input.
- **bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.
- **bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer and resets the output **bOccupied**.

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable **ipResultMessage**. The output is set to FALSE again as soon as **bBusy** switches to TRUE.
- **ipResultMessage**: Interface pointer (see [Error evaluation][574]) that can be used to obtain detailed information about the processing of the function block (see [runtime messages][561]). The interface pointer is valid after **bBusy** has changed from TRUE to FALSE.
- **bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.
- **bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.
- **bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.
- **bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.
- **nBrightnessLevel**: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.
PROPERTIES

All parameters that are written to the DALI control device via \textit{initialize} are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \textit{bQueryBrightness}.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \textit{bQueryOccupancy}.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the \textit{event priority [160]}.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the \textit{event priority [160]}.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>1 (high)</td>
<td>Setting the sensitivity of the occupancy sensor instance from 0 (max), 1 (high), 2 (low) to 3 (min).</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>
3.1.4.8 Osram

3.1.4.8.1 FB_DALI_Osram_Coupler_E_G2

The function block represents the DALI COUPLER E G2 DALI sensor from Osram. This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;

bError: Occurs when an error occurs during the execution of the function block.

ipResultMessage: Contains the message returned by the function block.

bInitializing: Indicates if the function block is initializing.

bReadingOccupancy: Indicates if the function block is reading the occupancy.
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES
All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input ( bQueryBrightness ).</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input ( bQueryOccupancy ).</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the ( \text{event_priority} ) ( 160 ).</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time ((0\ldots255 s)) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time ((0\ldots12750 ms)) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value ((0\ldots25 %)) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the ( \text{event_priority} ) ( 160 ).</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time ((0\ldots12750 ms)) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time ((1\ldots2538 s)) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time ((0\ldots255 s)) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

- **Development environment**
  - TwinCAT from v3.1.4024.20
- **Required PLC library**
  - Tc3_DALI from v3.7.0.0

### 3.1.4.8.2 FB_DALI_Osram_Coupler_HF_G2

The function block represents the DALI COUPLER HF G2 DALI sensor from Osram.
This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0). Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction

Part 304 (light sensors) - Introduction

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables \texttt{bQueryOccupancy} and \texttt{bQueryBrightness}.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

\begin{verbatim}
VAR_INPUT
 bInitialize             : BOOL := FALSE;
 nShortAddress          : BYTE := 0;
 nOptions               : DWORD := 0;
 bQueryOccupancy        : BOOL := FALSE;
 bQueryBrightness       : BOOL := FALSE;
 bCancelHoldTimerOccupancy : BOOL := FALSE;
\end{verbatim}

\textbf{bInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\textbf{nShortAddress}: Short address (0…63) of the DALI control device.

\textbf{nOptions}: Reserved for future extensions.

\textbf{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

\textbf{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

\textbf{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \texttt{bOccupied}.

\begin{verbatim}
VAR_OUTPUT
 bError                 : BOOL;
 ipResultMessage        : I_TcMessage;
 bInitializing          : BOOL;
 bReadingOccupancy      : BOOL;
 bReadingBrightness     : BOOL;
 bOccupied              : BOOL;
 nBrightnessLevel       : UINT;
\end{verbatim}

\textbf{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\textbf{ipResultMessage}: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\textbf{bInitializing}: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

\textbf{bReadingOccupancy}: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

\textbf{bReadingBrightness}: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.
**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

> This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

### PROPERTIES

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>
The function block represents the DALI COUPLER LS HIGHBAY G2 DALI sensor from Osram.

This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

! Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

! The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;
**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy:** The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.
### Name | Type | Access | Initial Value | Description
--- | --- | --- | --- | ---
bEnableBrightness | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input `bQueryBrightness`.

bEnableOccupancy | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input `bQueryOccupancy`.

eEventPriorityBrightness | E_DALIEventPriority | Get, Set | MiddleLow | Setting of the event priority [160].

nReportTimerBrightness | BYTE | Get, Set | 30 s | Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.

nDeadtimeTimerBrightness | UINT | Get, Set | 1500 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.

nHysteresisBrightness | BYTE | Get, Set | 5 % | Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.

nHysteresisMinBrightness | BYTE | Get, Set | 10 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.

eEventPriorityOccupancy | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].

nDeadtimeTimerOccupancy | UINT | Get, Set | 100 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.

nHoldTimerOccupancy | UINT | Get, Set | 900 s | Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.

nReportTimerOccupancy | BYTE | Get, Set | 60 s | Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.8.4 FB_DALI_Osram_Coupler_Multi3_G2

The function block represents the DALI COUPLER MULTI3 G2 DALI sensor from Osram.
This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0).
Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [▶ 313]
Part 304 (light sensors) - Introduction [▶ 327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables \texttt{bQueryOccupancy} and \texttt{bQueryBrightness}.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

\begin{Verbatim}
\textbf{VAR_INPUT}
\begin{verbatim}
\texttt{bInitialize} : BOOL := FALSE;
\texttt{nShortAddress} : BYTE := 0;
\texttt{nOptions} : DWORD := 0;
\texttt{bQueryOccupancy} : BOOL := FALSE;
\texttt{bQueryBrightness} : BOOL := FALSE;
\texttt{bCancelHoldTimerOccupancy} : BOOL := FALSE;
\end{verbatim}
\end{Verbatim}

\textit{bInitialize:} The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\textit{nShortAddress:} Short address (0…63) of the DALI control device.

\textit{nOptions:} Reserved for future extensions.

\textit{bQueryOccupancy:} The occupancy is queried immediately by a positive edge at this input.

\textit{bQueryBrightness:} The brightness is queried immediately by a positive edge at this input.

\textit{bCancelHoldTimerOccupancy:} A positive edge at this input prematurely terminates the Hold Timer and resets the output \texttt{bOccupied}.

\begin{Verbatim}
\textbf{VAR_OUTPUT}
\begin{verbatim}
\texttt{bError} : BOOL;
\texttt{ipResultMessage} : I_TcMessage;
\texttt{bInitializing} : BOOL;
\texttt{bReadingOccupancy} : BOOL;
\texttt{bReadingBrightness} : BOOL;
\texttt{bOccupied} : BOOL;
\texttt{nBrightnessLevel} : UINT;
\end{verbatim}
\end{Verbatim}

\textit{bError:} This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \texttt{ipResultMessage}. The output is set to FALSE again as soon as \texttt{bBusy} switches to TRUE.

\textit{ipResultMessage:} Interface pointer (see Error evaluation [▶ 574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [▶ 561]). The interface pointer is valid after \texttt{bBusy} has changed from TRUE to FALSE.

\textit{bInitializing:} The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

\textit{bReadingOccupancy:} The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

\textit{bReadingBrightness:} The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.
**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor. This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

### PROPERTIES

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bEnableBrightness</code></td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td><code>bEnableOccupancy</code></td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td><code>eEventPriorityBrightness</code></td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td><code>nReportTimerBrightness</code></td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td><code>nDeadtimeTimerBrightness</code></td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td><code>nHysteresisBrightness</code></td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td><code>nHysteresisMinBrightness</code></td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td><code>eEventPriorityOccupancy</code></td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td><code>nDeadtimeTimerOccupancy</code></td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td><code>nHoldTimerOccupancy</code></td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td><code>nReportTimerOccupancy</code></td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the DALI COUPLER PushB G2 DALI push button interface from Osram.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - [Introduction](#) 288

The existing parameters can be overwritten by changing the properties listed below.

The input variables bQueryInputValueN trigger the manual reading of the corresponding instance.

The properties bEnablePushButtonN make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at bInitialize and stored there.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

---

The function block cannot be used when using the KL6811.

---

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryInputValue1</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryInputValue2</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryInputValue3</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryInputValue4</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryInputValueN**: The state of the corresponding push button is queried immediately by a positive edge at this input.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue1</td>
<td>BOOL;</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

**PROPERTIES**

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>Push button input 1…4, which can be either enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000001</td>
<td>Setting of the event filter [1.288].</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DLLIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [1.160].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100…2000 ms) of repetitive events in the case of a long keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
</tbody>
</table>

**Requirements**

**Development environment**
TwinCAT from v3.1.4024.20

**Required PLC library**
Tc3_DALI from v3.7.0.0

---

**3.1.4.8.6 FB_DALI_Osram_LS_PD_CI_G2**

The function block represents the LS/PD CI G2 DALI sensor from Osram.

This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [1.313]

Part 304 (light sensors) - Introduction [1.327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

---

The function block cannot be used when using the KL6811.
VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError : BOOL;

ipResultMessage : I_TcMessage;

bInitializing : BOOL;

bReadingOccupancy : BOOL;

bReadingBrightness : BOOL;

bOccupied : BOOL;
nBrightnessLevel : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.8.7 FB_DALI_Osram_LS_PD_LI_G2

The function block represents the LS/PD LI G2 DALI sensor from Osram.
This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [☞ 313]
Part 304 (light sensors) - Introduction [☞ 327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL := FALSE;</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
<td>Interface pointer (see Error evaluation [☞ 574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [☞ 561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
<td>The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
<td>The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
<td>The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.</td>
</tr>
</tbody>
</table>
bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
3.1.4.8.8  **FB_DALI_Osram_LS_PD_LI_UF_G2**

The function block represents the LS/PD LI UF G2 DALI sensor from Osram. This outputs the measured brightness value (instance 1) and an occupancy signal (instance 0).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – **Introduction**

Part 304 (light sensors) – **Introduction**

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables `bQueryOccupancy` and `bQueryBrightness`.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output <code>bOccupied</code>.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
<td></td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT</td>
<td></td>
</tr>
</tbody>
</table>
**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy:** The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>
3.1.4.9 Steinel

3.1.4.9.1 FB_DALI_Steinel_ControlProDualHF

The function block represents the ControlPro Dual HF DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables bQueryOccupancy and bQueryBrightness.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
<td></td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT</td>
<td></td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor. This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
3.1.4.9.2  FB_DALI_Steinel_ControlProDualTech

The function block represents the ControlPro Dual Tech US Dali sensor from Steinel.

This outputs the measured brightness value (instance 0) and two independent occupancy signals (instance 1 and instance 2).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The parameters can be overwritten via the properties.

Readout can be triggered manually via the input variables bQueryOccupancy1, bQueryOccupancy2, and bQueryBrightness.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

- bInitialize : BOOL := FALSE;
- nShortAddress : BYTE := 0;
- nOptions : DWORD := 0;
- bQueryOccupancy1 : BOOL := FALSE;
- bQueryOccupancy2 : BOOL := FALSE;
- bQueryBrightness : BOOL := FALSE;
- bCancelHoldTimerOccupancy1 : BOOL := FALSE;
- bCancelHoldTimerOccupancy2 : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancyN: Immediate query of the occupancy at the respective instance.

bQueryBrightness: Immediate query of the brightness.

bCancelHoldTimerMotionN: A positive edge at this input prematurely terminates the Hold Timer of the respective instance and resets the output bOccupied.

VAR_OUTPUT

- bError : BOOL;
- ipResultMessage : I_TcMessage;
- bInitializing : BOOL;

bError: This indicates whether an error occurred during the operation.

ipResultMessage: Provides information about the result of the operation.

bInitializing: Indicates whether the initialization is currently in progress.
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancyN: The output is TRUE as long as values of the respective instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupiedN: If the output is TRUE, then occupancy has been detected in the detection range of the instance of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

Some parameters of the function block are executed as properties whose values can be changed if necessary (see Use of properties).

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Name | Type | Access | Initial value | Description
---|---|---|---|---
bEnableBrightness | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \texttt{bQueryBrightness}.
bEnableOccupancyN | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \texttt{bQueryOccupancy}.
eEventPriorityBrightness | E_DALIEventPriority | Get, Set | MiddleLow | Setting of the event priority \[160].
nReportTimerBrightness | BYTE | Get, Set | 30 s | Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.
nDeadtimeTimerBrightness | UINT | Get, Set | 1500 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
nHysteresisBrightness | BYTE | Get, Set | 20 % | Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.
nHysteresisMinBrightness | BYTE | Get, Set | 10 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.
eEventPriorityOccupancyN | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority \[160].
nDeadtimeTimerOccupancyN | UINT | Get, Set | 100 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
nHoldTimerOccupancyN | UINT | Get, Set | 900 s | Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.
nReportTimerOccupancyN | BYTE | Get, Set | 60 s | Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.
nSensitivityOccupancy | BYTE | Get, Set | 100 % | Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).
nDetectionRangeOccupancy | BYTE | Get, Set | 100 % | Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the ControlPro Dual US DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables boQueryOccupancy and boQueryBrightness.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;
**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

> This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via *bInitialize* are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \textit{bQueryBrightness}.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \textit{bQueryOccupancy}.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the ControlPro IR Quattro HD DALI sensor from Steinel. This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables bQueryOccupancy and bQueryBrightness.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

bInitailize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>lpResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
**bEnableBrightness** | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input *bQueryBrightness*.

**bEnableOccupancy** | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input *bQueryOccupancy*.

**eEventPriorityBrightness** | E_DALIEventPriority | Get, Set | MiddleLow | Setting of the event priority [160].

**nReportTimerBrightness** | BYTE | Get, Set | 30 s | Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.

**nDeadtimeTimerBrightness** | UINT | Get, Set | 1500 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.

**nHysteresisBrightness** | BYTE | Get, Set | 20 % | Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.

**nHysteresisMinBrightness** | BYTE | Get, Set | 10 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.

**eEventPriorityOccupancy** | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].

**nDeadtimeTimerOccupancy** | UINT | Get, Set | 100 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.

**nHoldTimerOccupancy** | UINT | Get, Set | 900 s | Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.

**nReportTimerOccupancy** | BYTE | Get, Set | 60 s | Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.

**nSensitivityOccupancy** | BYTE | Get, Set | 100 % | Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).

### Requirements

**Development environment**

TwinCAT from v3.1.4024.11

**Required PLC library**

Tc3_DALI from v3.5.0.0

#### 3.1.4.9.5 FB_DALI_Steinel_ControlProSingleUS

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>bool bError, char *ipErrorMessage, DWORD dwInstance, DWORD dwResultMessage</td>
</tr>
<tr>
<td>ShortAddress</td>
<td>char [20] String::cShortAddress</td>
</tr>
<tr>
<td>Options</td>
<td>DWORD dwOptions, DWORD dwUserOptions</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>bool bQueryOccupancy</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>bool bQueryBrightness</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>bool bCancelHoldTimerOccupancy</td>
</tr>
<tr>
<td>bError</td>
<td>bool bError</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>bool bReadingOccupancy</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>bool bReadingBrightness</td>
</tr>
<tr>
<td>bOccupied</td>
<td>bool bOccupied</td>
</tr>
<tr>
<td>brightnessLevel</td>
<td>UINT nBrightnessLevel</td>
</tr>
</tbody>
</table>
The function block represents the DALI ControlPro Single US DALI sensor from Steinel. This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [ 313]

Part 304 (light sensors) - Introduction [ 327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables $bQueryOccupancy$ and $bQueryBrightness$.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

### VAR_INPUT

- **bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- **nShortAddress**: Short address (0…63) of the DALI control device.

- **nOptions**: Reserved for future extensions.

- **bQueryOccupancy**: The occupancy is queried immediately by a positive edge at this input.

- **bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

- **bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer and resets the output $bOccupied$.

### VAR_OUTPUT

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable $ipResultMessage$. The output is set to FALSE again as soon as $bBusy$ switches to TRUE.

- **ipResultMessage**: Interface pointer (see Error evaluation [ 574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [ 561]). The interface pointer is valid after $bBusy$ has changed from TRUE to FALSE.

- **bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

- **bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.
**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor. This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

## PROPERTIES

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.4.9.6 **FB_DALI_Steinel_ControlProUS360**

The function block represents the ControlPro US 360 DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

- `bInitialize` : BOOL := FALSE;
- `nShortAddress` : BYTE := 0;
- `nOptions` : DWORD := 0;
- `bQueryOccupancy` : BOOL := FALSE;
- `bQueryBrightness` : BOOL := FALSE;
- `bcancelHoldTimerOccupancy` : BOOL := FALSE;

`bInitialize`: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

`nShortAddress`: Short address (0…63) of the DALI control device.

`nOptions`: Reserved for future extensions.

`bQueryOccupancy`: The occupancy is queried immediately by a positive edge at this input.

`bQueryBrightness`: The brightness is queried immediately by a positive edge at this input.

`bcancelHoldTimerOccupancy`: A positive edge at this input prematurely terminates the Hold Timer and resets the output `bOccupied`.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3.DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
3.1.4.9.7 FB_DALI_Steinel_DualLightSensor

The function block represents the Dual Light Sensor AP/UP DALI sensor from Steinel. The function block has two light sensors (instance 0 and instance 1). These two sensors each output a brightness value. Further information on the supported instance type can be found here:

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables bQueryBrightness1 and bQueryBrightness2.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryBrightness1 : BOOL := FALSE;
bQueryBrightness2 : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryBrightnessN: Immediate query of the brightness of the corresponding instance.

VAR_OUTPUT

bError : BOOL;

ipResultMessage : I_TcMessage;

bInitializing : BOOL;
bReadingBrightness1 : BOOL;
bReadingBrightness2 : BOOL;
nBrightnessLevel1 : UINT;
nBrightnessLevel2 : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValueBrightnessN: The output is TRUE as long as values of the respective instance are being read by the DALI control device.

nBrightnessLevelN: Measured brightness value of the respective light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightnessN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>eEventPriorityBrightnessN</td>
<td>E_DALI</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimeBrightnessN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the Hallway IPD DALI sensor from Steinel.

This outputs the measured brightness value (instance 0), three occupancy signals (instances 1 to 3), a temperature value (instance 4) and a relative humidity value (instance 5).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) - Introduction [313]
Part 304 (light sensors) - Introduction [327]

The DALI sensor contains three occupancy sensors. One occupancy sensor (instance 2) is aligned in the direction "S", the STEINEL lettering on the device, while the second occupancy sensor (instance 3) is aligned in the direction "L". The third occupancy sensor (instance 1) combines both sensors into a single instance.

Never activate all three instances at the same time, as this places an unnecessary load on the DALI bus. Either the two instances for direction "S" and "L" are active or the combined instance is (default setting). Thus either the properties \( nXyzDirectionS \) / \( nXyzDirectionL \) or the properties \( nXyzCombined \) should be set, but not all at the same time. Setting the properties \( nXyzCombined \) overwrites the values of \( nXyzDirectionS \) / \( nXyzDirectionL \).

The temperature measurement and humidity measurement are executed as Generic Inputs. These values can only be queried via a positive edge at the inputs \( bQueryTemperature \) and \( bQueryHumidity \).

For the other instances, readout can also be triggered manually via the inputs \( bQueryBrightness \), \( bQueryOccupancyCombined \), \( bQueryOccupancyDirectionS \) and \( bQueryOccupancyDirectionL \).

The existing parameters can be overwritten by changing the properties listed below.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>0;</td>
</tr>
<tr>
<td>bQueryOccupancyCombined</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bQueryOccupancyDirectionS</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
</tbody>
</table>

The function block cannot be used when using the KL6811.
**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryOccupancyCombined**: positive edge at this input immediately queries the value of the combined occupancy sensor.

**bQueryOccupancyDirectionS**: The value of the occupancy sensor for the direction "S" is queried immediately by a positive edge at this input.

**bQueryOccupancyDirectionL**: A positive edge at this input immediately queries the value of the occupancy sensor for direction "L".

**bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

**bQueryTemperature**: The temperature is queried immediately by a positive edge at this input. The DALI device does not support transmission of the temperature per event.

**bQueryHumidity**: The relative humidity is queried immediately by a positive edge at this input. The DALI device does not support transmission of relative humidity per event.

**bCancelHoldTimerOccupancyCombined**: A positive edge at this input prematurely terminates the Hold Timer for the combined occupancy sensor prematurely and resets the output \( b\text{OccupiedCombined} \).

**bCancelHoldTimerOccupancyDirectionS**: A positive edge at this input prematurely terminates the Hold Timer of the occupancy sensor for direction "S" and resets the output \( b\text{OccupiedDirectionS} \).

**bCancelHoldTimerOccupancyDirectionL**: A positive edge at this input prematurely terminates the Hold Timer of the occupancy sensor for direction "L" prematurely and resets the output \( b\text{OccupiedDirectionL} \).

**VAR_OUTPUT**

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \( ip\text{ResultMessage} \). The output is set to FALSE again as soon as \( b\text{Busy} \) switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \( b\text{Busy} \) has changed from TRUE to FALSE.

**bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancyCombined**: The output is TRUE as long as values of the instance of the combined motion sensor are being read by the DALI control device.
bReadingOccupancyDirectionS: The output is TRUE as long as values of the instance of the motion sensor for the direction "S" are being read by the DALI control device.

bReadingOccupancyDirectionL: The output is TRUE as long as values of the instance of the motion sensor for the direction "L" are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bReadingTemperature: The output is TRUE as long as values of the instance of the temperature sensor are being read by the DALI control device.

bReadingHumidity: The output is TRUE as long as values of the humidity sensor instance are read by the DALI controller.

bOccupiedCombined: If the output is TRUE, then occupancy has been detected in the detection range of the combined occupancy sensor.

bOccupiedDirectionS: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor for the direction "S".

bOccupiedDirectionL: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor for the direction "L".

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

fTemperature: Measured temperature value (-5.0...60.0 °C) of the temperature sensor.

nHumidity: Measured value of the relative humidity (0...100 %).

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy Combined</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>A TRUE at this property enables the instance for the combined occupancy sensor. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancyCombined</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy DirectionS</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for motion detection in direction &quot;S&quot;. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancyDirectionS</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy DirectionL</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for motion detection in direction &quot;L&quot;. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancyDirectionL</code>.</td>
</tr>
<tr>
<td>bEnableTemperature</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for temperature measurement. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input <code>bQueryTemperature</code>.</td>
</tr>
<tr>
<td>bEnableHumidity</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>A TRUE on this property enables the instance to measure relative humidity. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input <code>bQueryHumidity</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, set</td>
<td>20 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Access</td>
<td>Initial value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------------</td>
<td>---------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancyCombined</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancyCombined</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nDetectionRangeOccupancyCombined</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range). Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</td>
</tr>
<tr>
<td>nHoldTimerOccupancyCombined</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancyCombined</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancyCombined</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity). Setting this property overwrites nSensitivityOccupancyDirectionS and nSensitivityOccupancyDirectionL.</td>
</tr>
<tr>
<td>eEventPriorityOccupancyDirectionS</td>
<td>E_DALIEvEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancyDirectionS</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nDetectionRangeOccupancyDirectionS</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
<tr>
<td>nHoldTimerOccupancyDirectionS</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancyDirectionS</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancyDirectionS</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>eEventPriorityOccupancyDirectionL</td>
<td>E_DALIEvEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancyDirectionL</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
</tbody>
</table>
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.9.9 FB_DALI_Steinel_Hallway_IPD_ECO

The function block represents the Hallway IPD ECO DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and three occupancy signals (instances 1 to 3).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) - Introduction [313]

Part 304 (light sensors) - Introduction [327]

The DALI sensor contains three occupancy sensors. One occupancy sensor (instance 2) is aligned in the direction "S", the STEINEL lettering on the device, while the second occupancy sensor (instance 3) is aligned in the direction "L". The third occupancy sensor (instance 1) combines both sensors into a single instance.

Never activate all three instances at the same time, as this places an unnecessary load on the DALI bus. Either the two instances for direction "S" and "L" are active or the combined instance is (default setting). Thus either the properties `nXyzDirectionS / nXyzDirectionL` or the properties `nXyzCombined` should be set, but not all at the same time. Setting the properties `nXyzCombined` overwrites the values of `nXyzDirectionS / nXyzDirectionL`.

The readout can be triggered manually via the input variables `bQueryBrightness`, `bQueryOccupancyCombined`, `bQueryOccupancyDirectionS`, and `bQueryOccupancyDirectionL`. 
The existing parameters can be overwritten by changing the properties listed below.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryOccupancyCombined</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryOccupancyDirectionS</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryOccupancyDirectionL</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancyCombined</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancyDirectionS</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancyDirectionL</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**nOptions**: Reserved for future extensions.

**bQueryOccupancyCombined**: positive edge at this input immediately queries the value of the combined occupancy sensor.

**bQueryOccupancyDirectionS**: The value of the occupancy sensor for the direction "S" is queried immediately by positive edge at this input.

**bQueryOccupancyDirectionL**: A positive edge at this input immediately queries the value of the occupancy sensor for direction "L".

**bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

**bCancelHoldTimerOccupancyCombined**: A positive edge at this input prematurely terminates the Hold Timer for the combined occupancy sensor prematurely and resets the output bOccupiedCombined.

**bCancelHoldTimerOccupancyDirectionS**: A positive edge at this input prematurely terminates the Hold Timer of the occupancy sensor for direction "S" and resets the output bOccupiedDirectionS.

**bCancelHoldTimerOccupancyDirectionL**: A positive edge at this input prematurely terminates the Hold Timer of the occupancy sensor for direction "L" prematurely and resets the output bOccupiedDirectionL.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancyCombined</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancyDirectionS</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancyDirectionL</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupiedCombined</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupiedDirectionS</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupiedDirectionL</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancyCombined: The output is TRUE as long as values of the instance of the combined motion sensor are being read by the DALI control device.

bReadingOccupancyDirectionS: The output is TRUE as long as values of the instance of the motion sensor for the direction "S" are being read by the DALI control device.

bReadingOccupancyDirectionL: The output is TRUE as long as values of the instance of the motion sensor for the direction "L" are being read by the DALI control device.

bReadingHumidity: The output is TRUE as long as values of the humidity sensor instance are read by the DALI controller.

bOccupiedCombined: If the output is TRUE, then occupancy has been detected in the detection range of the combined occupancy sensor.

bOccupiedDirectionS: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor for the direction "S".

bOccupiedDirectionL: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor for the direction "L".

nBrightnessLevel: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <em>bQueryBrightness</em>.</td>
</tr>
<tr>
<td>bEnableOccupancy Combined</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>A TRUE at this property enables the instance for the combined occupancy sensor. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <em>bQueryOccupancyCombined</em>.</td>
</tr>
<tr>
<td>bEnableOccupancy DirectionS</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for motion detection in direction &quot;S&quot;. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <em>bQueryOccupancyDirectionS</em>.</td>
</tr>
<tr>
<td>bEnableOccupancy DirectionL</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for motion detection in direction &quot;L&quot;. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <em>bQueryOccupancyDirectionL</em>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, set</td>
<td>20 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancyCombined</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancyCombined</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Access</td>
<td>Initial value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>--------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>nDetectionRangeOccupancyCombined</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range). <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nHoldTimerOccupancyCombined</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor. <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nReportTimerOccupancyCombined</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime. <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nSensitivityOccupancyCombined</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity). <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>eEventPriorityOccupancyDirectionS</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancyDirectionS</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nDetectionRangeOccupancyDirectionS</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
<tr>
<td>nHoldTimerOccupancyDirectionS</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor. <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nReportTimerOccupancyDirectionS</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime. <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nSensitivityOccupancyDirectionS</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity). <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>eEventPriorityOccupancyDirectionL</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancyDirectionL</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nDetectionRangeOccupancyDirectionL</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
<tr>
<td>nHoldTimerOccupancyDirectionL</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor. <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nReportTimerOccupancyDirectionL</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime. <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
<tr>
<td>nSensitivityOccupancyDirectionL</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity). <strong>Setting this property overwrites nDetectionRangeOccupancyDirectionS and nDetectionRangeOccupancyDirectionL.</strong></td>
</tr>
</tbody>
</table>
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
ipDALICommunication | I_DALICommunication | Get, Set | 0 | Interface pointer to the communication block (e.g. FB_KL6821Communication [343]).

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.9.10 FB_DALI_Steinel_HF3360

The function block represents the HF 3360 AP/UP DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

- Part 303 (occupancy sensors) – [Introduction](#)
- Part 304 (light sensors) – [Introduction](#)

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

#### VAR_INPUT

- `bInitialize`: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
- `nShortAddress`: Short address (0…63) of the DALI control device.
- `nOptions`: Reserved for future extensions.
- `bQueryOccupancy`: The occupancy is queried immediately by a positive edge at this input.
- `bQueryBrightness`: The brightness is queried immediately by a positive edge at this input.
bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the HF 360 DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables $bQueryOccupancy$ and $bQueryBrightness$.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

$bInitialize$: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

$nShortAddress$: Short address (0…63) of the DALI control device.

$nOptions$: Reserved for future extensions.

$bQueryOccupancy$: The occupancy is queried immediately by a positive edge at this input.

$bQueryBrightness$: The brightness is queried immediately by a positive edge at this input.

$bCancelHoldTimerOccupancy$: A positive edge at this input prematurely terminates the Hold Timer and resets the output $bOccupied$.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
The function block represents the HF 360 II IPD DALI sensor from Steinel.

This outputs the measured brightness value (instance 0), a occupancy signal (instance 1), a temperature value (instance 2) and a value for the relative humidity (instance 3).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) - Introduction [313]
Part 304 (light sensors) - Introduction [327]

The temperature measurement and humidity measurement are executed as Generic Inputs. These values can only be queried via a positive edge at the inputs bQueryTemperature and bQueryHumidity.

For the other instances, the readout can also be triggered manually via the inputs bQueryBrightness and bQueryOccupancy.

The existing parameters can be overwritten by changing the properties listed below.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bQueryTemperature</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bQueryHumidity</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bReadingHumidity</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bReadingTemperature</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL</td>
<td>FALSE;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT</td>
<td>0;</td>
</tr>
<tr>
<td>nTemperature</td>
<td>LREAL</td>
<td>0;</td>
</tr>
<tr>
<td>nHumidity</td>
<td>BYTE</td>
<td>0;</td>
</tr>
</tbody>
</table>

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The value of the occupancy sensor is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.
**bQueryTemperature**: The temperature is queried immediately by a positive edge at this input. The DALI device does not support transmission of the temperature per event.

**bQueryHumidity**: The relative humidity is queried immediately by a positive edge at this input. The DALI device does not support transmission of relative humidity per event.

**bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer for the combined occupancy sensor prematurely and rests the output **bOccupied**.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingTemperature</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingHumidity</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
<tr>
<td>fTemperature</td>
<td>LREAL;</td>
</tr>
<tr>
<td>nHumidity</td>
<td>BYTE;</td>
</tr>
</tbody>
</table>

**bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable **ipResultMessage**. The output is set to FALSE again as soon as **bBusy** switches to TRUE.

**ipResultMessage**: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after **bBusy** has changed from TRUE to FALSE.

**bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bReadingTemperature**: The output is TRUE as long as values of the instance of the temperature sensor are being read by the DALI control device.

**bReadingHumidity**: The output is TRUE as long as values of the humidity sensor instance are read by the DALI controller.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**fTemperature**: Measured temperature value (-5.0...60.0 °C) of the temperature sensor.

**nHumidity**: Measured value of the relative humidity (0...100 %).

### PROPERTIES

All parameters that are written to the DALI control device via **bInitialize** are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <em>bQueryBrightness</em>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>A TRUE at this property enables the instance for the occupancy sensor. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <em>bQueryOccupancy</em>.</td>
</tr>
<tr>
<td>bEnableTemperature</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the instance for temperature measurement. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input <em>bQueryTemperature</em>.</td>
</tr>
<tr>
<td>bEnableHumidity</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>A TRUE on this property enables the instance to measure relative humidity. FALSE at this property causes the instance not to be configured during initialization. The actual value of the instance can only be queried via the input <em>bQueryHumidity</em>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEv</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
</tbody>
</table>
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.9.13 FB_DALI_Steinel_HF360_II_IPD_ECO

The function block represents the HF 360 II IPD ECO DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – [Introduction](#)

Part 304 (light sensors) - [Introduction](#)

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

- **Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.**

- **The function block cannot be used when using the KL6811.**

#### VAR_INPUT

<table>
<thead>
<tr>
<th></th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

- **bInitialize:** The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- **nShortAddress:** Short address (0…63) of the DALI control device.
nOptions: Reserved for future extensions.

bQueryOccupancy: The value of the occupancy sensor is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer for the combined occupancy sensor prematurely and rests the output bOccupied.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>lpResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable lpResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

lpResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bReadingTemperature: The output is TRUE as long as values of the instance of the temperature sensor are being read by the DALI control device.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>A TRUE at this property enables the instance for the occupancy sensor. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, set</td>
<td>20 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nDetectionRangeOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the detection range of the occupancy sensor from 0 % (small detection range) to 100 % (large detection range).</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
<tr>
<td>ipDALICommunication</td>
<td>I_DALICommunicati on</td>
<td>Get, Set</td>
<td>0</td>
<td>Interface pointer to the communication block (e.g. FB_KL6821Communication [343]).</td>
</tr>
</tbody>
</table>
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4024.25</td>
<td>Tc3_DALI v3.10.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.9.14  FB_DALI_Steinel_IRMicro

The function block represents the IR Micro DALI sensor from Steinel. This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

- Part 303 (occupancy sensors) – [Introduction](#)
- Part 304 (light sensors) - [Introduction](#)

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.
- The function block cannot be used when using the KL6811.

#### VAR_INPUT

- `bInitialize`: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
- `nShortAddress`: Short address (0…63) of the DALI control device.
- `nOptions`: Reserved for future extensions.
- `bQueryOccupancy`: The occupancy is queried immediately by a positive edge at this input.
- `bQueryBrightness`: The brightness is queried immediately by a positive edge at this input.
- `bCancelHoldTimerOccupancy`: A positive edge at this input prematurely terminates the Hold Timer and resets the output `bOccupied`.

#### VAR_OUTPUT

- `bError`:  
- `ipResultMessage`:  
- `bInitializing`: 
**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingOccupancy:** The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

**bReadingBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

> This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via *bInitialize* are available as properties.
### Requirements

**Development environment**

TwinCAT from v3.1.4024.11

**Required PLC library**

Tc3_DALI from v3.5.0.0

### 3.1.4.9.15  FB_DALI_Steinel_IRQuattroMicro

**FB_DALI_Steinel_IRQuattroMicro**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td></td>
</tr>
<tr>
<td>nShortAddress</td>
<td></td>
</tr>
<tr>
<td>nOptions</td>
<td></td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td></td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td></td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td></td>
</tr>
<tr>
<td>bCancelHoldTimerBrightness</td>
<td></td>
</tr>
<tr>
<td>bEnableBrightness</td>
<td></td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td></td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td></td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td></td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td></td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td></td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td></td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td></td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td></td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td></td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td></td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td></td>
</tr>
</tbody>
</table>

**FB_DALI_Steinel_IRQuattroMicro**

- **nShortAddress**: BYTE
- **nOptions**: DWORD
- **bQueryOccupancy**: BOOL
- **bQueryBrightness**: BOOL
- **bCancelHoldTimerOccupancy**: BOOL
- **bCancelHoldTimerBrightness**: BOOL
- **bEnableBrightness**: BOOL
- **bEnableOccupancy**: BOOL
- **eEventPriorityBrightness**: E_DALIEventPriority
- **nReportTimerBrightness**: BYTE
- **nDeadtimeTimerBrightness**: UINT
- **nHysteresisBrightness**: BYTE
- **nHysteresisMinBrightness**: BYTE
- **eEventPriorityOccupancy**: E_DALIEventPriority
- **nReportTimerOccupancy**: BYTE
- **nDeadtimeTimerOccupancy**: UINT
- **nHoldTimerOccupancy**: UINT
- **nSensitivityOccupancy**: BYTE
- **bError**: BOOL
- **bInitializing**: BOOL
- **bReadingOccupancy**: BOOL
- **bReadingBrightness**: BOOL
- **bOccupied**: BOOL
- **bOccupiedLevel**: UINT

**Version: 1.10**

TE1000
The function block represents the IR Quattro Micro DALI sensor from Steinel. This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables bQueryOccupancy and bQueryBrightness.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.
**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

> This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.4.9.16   FB_DALI_Steinel_IRQuattroSlimXS

The function block represents the IR Quattro Slim XS DALI sensor from Steinel. This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables \texttt{bQueryOccupancy} and \texttt{bQueryBrightness}.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

VAR_INPUT

\begin{verbatim}
blInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;
\end{verbatim}

\texttt{blInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\texttt{nShortAddress}: Short address (0…63) of the DALI control device.

\texttt{nOptions}: Reserved for future extensions.

\texttt{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

\texttt{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

\texttt{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \texttt{bOccupied}. 

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via `bInitialize` are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.4.9.17 **FB_DALI_Steinel_IS345**
The function block represents the IS 345 AP/UP DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction

Part 304 (light sensors) - Introduction

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

- `bInitialize`: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- `nShortAddress`: Short address (0…63) of the DALI control device.

- `nOptions`: Reserved for future extensions.

- `bQueryOccupancy`: The occupancy is queried immediately by a positive edge at this input.

- `bQueryBrightness`: The brightness is queried immediately by a positive edge at this input.

- `bCancelHoldTimerOccupancy`: A positive edge at this input prematurely terminates the Hold Timer and resets the output `bOccupied`.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT;</td>
</tr>
</tbody>
</table>

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

- `ipResultMessage`: Interface pointer (see Error evaluation) that can be used to obtain detailed information about the processing of the function block (see runtime messages). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

- `bInitializing`: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

- `bReadingOccupancy`: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.
**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor. This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

### PROPERTIES

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3.DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.4.9.18  FB.DALI_Steinel_IS345MX

The function block represents the IS 345 MX AP/UP DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables \texttt{bQueryOccupancy} and \texttt{bQueryBrightness}.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

\begin{verbatim}
  bInitialize : BOOL := FALSE;
  nShortAddress : BYTE := 0;
  nOptions : DWORD := 0;
  bQueryOccupancy : BOOL := FALSE;
  bQueryBrightness : BOOL := FALSE;
  bCancelHoldTimerOccupancy : BOOL := FALSE;
\end{verbatim}

- \texttt{bInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- \texttt{nShortAddress}: Short address (0…63) of the DALI control device.

- \texttt{nOptions}: Reserved for future extensions.

- \texttt{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.

- \texttt{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.

- \texttt{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \texttt{bOccupied}.

VAR_OUTPUT

\begin{verbatim}
  bError : BOOL;
  ipResultMessage : I_TcMessage;
  bInitializing : BOOL;
\end{verbatim}
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
\(b\text{EnableBrightness}\) | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \(b\text{QueryBrightness}\).
\(b\text{EnableOccupancy}\) | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \(b\text{QueryOccupancy}\).
\(e\text{EventPriorityBrightness}\) | E_DALIEventPriority | Get, Set | MiddleLow | Setting of the event priority [160].
\(n\text{ReportTimerBrightness}\) | BYTE | Get, Set | 30 s | Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.
\(n\text{DeadtimeTimerBrightness}\) | UINT | Get, Set | 1500 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
\(n\text{HysteresisBrightness}\) | BYTE | Get, Set | 20 % | Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.
\(n\text{HysteresisMinBrightness}\) | BYTE | Get, Set | 10 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.
\(e\text{EventPriorityOccupancy}\) | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
\(n\text{DeadtimeTimerOccupancy}\) | UINT | Get, Set | 100 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
\(n\text{HoldTimerOccupancy}\) | UINT | Get, Set | 900 s | Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.
\(n\text{ReportTimerOccupancy}\) | BYTE | Get, Set | 60 s | Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.
\(n\text{SensitivityOccupancy}\) | BYTE | Get, Set | 100 % | Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.9.19 FB_DALI_Steinel_IS3180

![FB_DALI_Steinel_IS3180](image-url)
The function block represents the IS 3180 AP/UP DALI sensor from Steinel. This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1). Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables \textit{bQueryOccupancy} and \textit{bQueryBrightness}.

\begin{itemize}
  \item Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.
  \item The function block cannot be used when using the KL6811.
\end{itemize}

\begin{varinput}
  \textit{bInitialize}: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
  \textit{nShortAddress}: Short address (0…63) of the DALI control device.
  \textit{nOptions}: Reserved for future extensions.
  \textit{bQueryOccupancy}: The occupancy is queried immediately by a positive edge at this input.
  \textit{bQueryBrightness}: The brightness is queried immediately by a positive edge at this input.
  \textit{bCancelHoldTimerOccupancy}: A positive edge at this input prematurely terminates the Hold Timer and resets the output \textit{bOccupied}.
\end{varinput}

\begin{varoutput}
  \textit{bError}: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.
  \textit{ipResultMessage}: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.
  \textit{bInitializing}: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.
  \textit{bReadingOccupancy}: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.
\end{varoutput}
bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.4.9.20  **FB_DALI_Steinel_IS3360**

The function block represents the IS 3360 AP/UP DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

1. Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

2. The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>Configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output <code>bOccupied</code>.</td>
</tr>
</tbody>
</table>
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input ( b\text{QueryBrightness} ).</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input ( b\text{QueryOccupancy} ).</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority ( \text{[160]} ).</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time ( (0\ldots255\text{ s}) ) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time ( (0\ldots12750\text{ ms}) ) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value ( (0\ldots25%) ) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority ( \text{[160]} ).</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time ( (0\ldots12750\text{ ms}) ) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time ( (1\ldots2538\text{ s}) ) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time ( (0\ldots255\text{ s}) ) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**3.1.4.9.21 FB.DALI_Steinel_IS3360MX**

![FB.DALI_Steinel_IS3360HX](image)

**TE1000**

Version: 1.10

487
The function block represents the IS 3360 MX AP/UP DALI sensor from Steinel.

This outputs the measured brightness value (instance 0) and an occupancy signal (instance 1).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) - Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables bQueryOccupancy and bQueryBrightness.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

**VAR_INPUT**

```
VAR_INPUT
bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;
```

- **bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- **nShortAddress**: Short address (0…63) of the DALI control device.

- **nOptions**: Reserved for future extensions.

- **bQueryOccupancy**: The occupancy is queried immediately by a positive edge at this input.

- **bQueryBrightness**: The brightness is queried immediately by a positive edge at this input.

- **bCancelHoldTimerOccupancy**: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

**VAR_OUTPUT**

```
VAR_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;
```

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

- **bInitializing**: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

- **bReadingOccupancy**: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.
**bReadingBrightness**: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied**: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel**: Measured brightness value of the light sensor.

- This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryBrightness</code>.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input <code>bQueryOccupancy</code>.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0...255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 %</td>
<td>Value (0...25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>100 %</td>
<td>Setting of the sensitivity of the occupancy sensor from 0 % (low sensitivity) to 100 % (high sensitivity).</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.1.4.10 Sunricher

3.1.4.10.1 FB_DALI_Sunricher_PushButtonCoupler

The function block represents the Push Button Coupler DALI push button interface from Sunricher.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer’s device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]

The existing parameters can be overwritten by changing the properties listed below.

The input variables bQueryInputValueN trigger the immediate reading of the corresponding instance.

The properties bEnablePushButtonN make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at bInitialize and stored there.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

| bInitialize | BOOL := FALSE; |
| nShortAddress | BYTE := 0; |
| nOptions | DWORD := 0; |
| bQueryInputValue1 | BOOL := FALSE; |
| bQueryInputValue2 | BOOL := FALSE; |
| bQueryInputValue3 | BOOL := FALSE; |
| bQueryInputValue4 | BOOL := FALSE; |

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
**nShortAddress:** Short address (0…63) of the DALI control device.

**nOptions:** Reserved for future extensions.

**bQueryInputValueN:** The state of the corresponding push button is queried immediately by a positive edge at this input.

**VAR_OUTPUT**

```plaintext
bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingInputValue1 : BOOL;
bReadingInputValue2 : BOOL;
bReadingInputValue3 : BOOL;
bReadingInputValue4 : BOOL;
nPushButton1Event : WORD;
nPushButton2Event : WORD;
nPushButton3Event : WORD;
nPushButton4Event : WORD;
```

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.

**ipResultMessage:** Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after `bBusy` has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bReadingValueN:** The outputs are set as soon as the manual reading of the corresponding input has been triggered.

**nPushButtonNEvent:** The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block **FB_DALI_ToPushButtonState [542]** can be used to convert the state of a push button into a BOOL variable.

**PROPERTIES**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Requirements

#### Development environment

- TwinCAT from v3.1.4024.25

<table>
<thead>
<tr>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI from v3.9.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.11 Theben

#### 3.1.4.11.1 FB_DALI_Theben_PlanoSpot_360

The function block represents the PlanoSpot 360 DALI sensor from Theben.
This outputs two occupancy signals (instances 0 and 1), four measured brightness values (instances 2 to 5) and eight push button signals (instances 6 to 13).

The push button instances can be used via an infrared remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]
Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) – Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables bQueryOccupancyStandard, bQueryOccupancyReduced, bQueryBrightnessIntegral, bQueryBrightnessInside, bQueryBrightnessCenter and bQueryBrightnessWindow.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancyN: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightnessN: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancyN: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError
ipResultMessage: I_TcMessage;
Initializing: BOOL;
bReadingOccupancyStandard: BOOL;
bReadingOccupancyReduced: BOOL;
bReadingBrightnessIntegral: BOOL;
bReadingBrightnessInside: BOOL;
bReadingBrightnessCenter: BOOL;
bReadingBrightnessWindow: BOOL;
bOccupiedStandard: BOOL;
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancyN: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightnessN: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupiedN: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevelN: Measured brightness value of the light sensor.

These measured values must be compared with the measured reference values at the place of operation of the DALI control device.

nPushButtonEventN: The output outputs the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightnessN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightnessN.</td>
</tr>
<tr>
<td>bEnableOccupancyN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancyN.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input that can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBrightnessN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nRoomCorrectionFactorN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30</td>
<td>The room correction factor (5…200) specifies the difference in brightness measurement on the ceiling and at the workplace. It can be calculated using a luxmeter with the following equation: [ \text{RoomCorrectionFactor} = \frac{\text{Brightness under ceiling}}{\text{Brightness on work surface}} ] The value can also be determined by a remote control and with an appropriate app. The preset value of 0.3 (30) is suitable for most applications.</td>
</tr>
<tr>
<td>nReportTimerBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimeBrightnessN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>163</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancyN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority.</td>
</tr>
<tr>
<td>nDeadtimeTimeOccupancyN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancyN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancyN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Access</td>
<td>Initial value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>---------</td>
<td>---------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>nSensitivityOcc upancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>3</td>
<td>Setting of the sensitivity of both occupancy sensor instances from 0 (low sensitivity) to 5 (high sensitivity).</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DALIE ventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [1.160].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time [0, tDoubleMin...2000 ms] to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100...2000 ms) of repetitive events in the case of a long keystroke. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.</td>
</tr>
</tbody>
</table>

### Requirements

#### Development environment
- TwinCAT from v3.1.4.11
- Tc3_DALI from v3.5.0.0

#### 3.1.4.11.2 FB_DALI_Theben_thePassa_P360

```c
FB_DALI_Theben_thePassa_P360

BOOL bInitialize
BYTE bShortAddress
DWORD bOptions
BOOL bQueryOccupancyOverall
BOOL bQueryOccupancyZone1
BOOL bQueryOccupancyZone2
BOOL bQueryBrightnessZone1
BOOL bQueryBrightnessZone2
BOOL bCancelHoldTimerOccupancyOverall
BOOL bCancelHoldTimerOccupancyZone1
BOOL bCancelHoldTimerOccupancyZone2
BOOL bError
Tc3_EventLogger bIpResultMessage
```

The function block represents thePassa P360 DALI sensor from Theben.

This outputs three occupancy signals (instances 0 to 2), three measured brightness values (instances 3 to 5) and eight push button signals (instances 6 to 13).

The push button instances can be used via an infrared remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.
Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [p. 288]

Part 303 (occupancy sensors) – Introduction [p. 313]

Part 304 (light sensors) – Introduction [p. 327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered immediately via the input variables \( bQueryOccupancyOverall \), \( bQueryOccupancyZone1 \), \( bQueryOccupancyZone2 \), \( bQueryBrightnessZone1and2 \), \( bQueryBrightnessZone1 \), \( bQueryBrightnessZone2 \).

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

---

**VAR_INPUT**

- \( bInitialize \) : BOOL := FALSE;
- \( nShortAddress \) : BYTE := 0;
- \( nOptions \) : DWORD := 0;
- \( bQueryOccupancyOverall \) : BOOL := FALSE;
- \( bQueryOccupancyZone1 \) : BOOL := FALSE;
- \( bQueryOccupancyZone2 \) : BOOL := FALSE;
- \( bQueryBrightnessZone1and2 \) : BOOL := FALSE;
- \( bQueryBrightnessZone1 \) : BOOL := FALSE;
- \( bQueryBrightnessZone2 \) : BOOL := FALSE;
- \( bCancelHoldTimerOccupancyOverall \) : BOOL := FALSE;
- \( bCancelHoldTimerOccupancyZone1 \) : BOOL := FALSE;
- \( bCancelHoldTimerOccupancyZone2 \) : BOOL := FALSE;

\( bInitialize \): The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

\( nShortAddress \): Short address (0…63) of the DALI control device.

\( nOptions \): Reserved for future extensions.

\( bQueryOccupancyN \): The occupancy is queried immediately by a positive edge at this input.

\( bQueryBrightnessN \): The brightness is queried immediately by a positive edge at this input.

\( bCancelHoldTimerOccupancyN \): A positive edge at this input prematurely terminates the Hold Timer and resets the output \( bOccupied \).

---

**VAR_OUTPUT**

- \( bError \) : BOOL;
- \( ipResultMessage \) : I_TcMessage;
- \( bInitializing \) : BOOL;
- \( bReadingOccupancyOverall \) : BOOL;
- \( bReadingOccupancyZone1 \) : BOOL;
- \( bReadingOccupancyZone2 \) : BOOL;
- \( bReadingBrightnessZone1and2 \) : BOOL;
- \( bReadingBrightnessZone1 \) : BOOL;
- \( bReadingBrightnessZone2 \) : BOOL;
- \( bOccupiedOverall \) : BOOL;
- \( bOccupiedZone1 \) : BOOL;
- \( bOccupiedZone2 \) : BOOL;
- \( nBrightnessLevelZone1and2 \) : UINT;
- \( nBrightnessLevelZone1 \) : UINT;
- \( nBrightnessLevelZone2 \) : UINT;
- \( nPushButtonEvent1On \) : WORD;
- \( nPushButtonEvent1Off \) : WORD;
- \( nPushButtonEvent2On \) : WORD;
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancyN: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightnessN: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupiedN: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevelN: Measured brightness value of the light sensor.

These measured values must be compared with the measured reference values at the place of operation of the DALI control device.

nPPushButtonEventN: The output outputs the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightnessN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightnessN.</td>
</tr>
<tr>
<td>bEnableOccupancyN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancyN.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input that can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBrightnessN</td>
<td>E_DALI</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nRoomCorrectionFactorN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30</td>
<td>The room correction factor (5…200) specifies the difference in brightness measurement on the ceiling and at the workplace. It can be calculated using a luxmeter with the following equation: $\text{RoomCorrectionFactor} = \frac{\text{Brightness under ceiling}}{\text{Brightness on work surface}}$. The value can also be determined by a remote control and with an appropriate app. The preset value of 0.3 (30) is suitable for most applications.</td>
</tr>
<tr>
<td>nReportTimerBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimeBrightnessN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>163</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancyN</td>
<td>E_DALI</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimeOccupancyN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancyN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancyN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>3</td>
<td>Setting of the sensitivity of both occupancy sensor instances from 0 (low sensitivity) to 5 (high sensitivity).</td>
</tr>
</tbody>
</table>
### Name | Type | Access | Initial value | Description
---|---|---|---|---
eEventPriorityPushButtonN | E_DALI EventPriority | Get, Set | Middle | Setting of the event priority \[1.10\].
nDoubleTimerPushButtonN | UINT | Get, Set | 0 ms | Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.
nRepeatTimerPushButtonN | UINT | Get, Set | 160 ms | Interval (100…2000 ms) of repetitive events in the case of a long keystroke. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.
nShortTimerPushButtonN | UINT | Get, Set | 500 ms | Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.
nStuckTimerPushButtonN | BYTE | Get, Set | 20 s | Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.

### Requirements

**Development environment**

- TwinCAT from v3.1.4024.11

**Required PLC library**

- Tc3_DALI from v3.5.0.0

### 3.1.4.11.3 FB_DALI_Theben_theRonda_P360

The function block represents the Ronda P360 DALI sensor from Theben.

This outputs one occupancy signal (instance 0), four measured brightness values (instances 1 to 4) and eight push-button messages (instances 5 to 12).

The push button instances can be used via an infrared remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer’s device documentation.

Further information on the supported instance types can be found here:

- Part 301 (push buttons) - [Introduction \[288\]]
Part 303 (occupancy sensors) – Introduction [313]

Part 304 (light sensors) – Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy`, `bQueryBrightnessIntegral`, `bQueryBrightnessInside`, `bQueryBrightnessCenter`, and `bQueryBrightnessWindow`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

### VAR_INPUT

- `bInitialize`: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.
- `nShortAddress`: Short address (0…63) of the DALI control device.
- `nOptions`: Reserved for future extensions.
- `bQueryOccupancy`: The occupancy is queried immediately by a positive edge at this input.
- `bQueryBrightnessN`: The brightness is queried immediately by a positive edge at this input.
- `bCancelHoldTimerOccupancy`: A positive edge at this input prematurely terminates the Hold Timer and resets the output `bOccupied`.

### VAR_OUTPUT

- `bError`: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable `ipResultMessage`. The output is set to FALSE again as soon as `bBusy` switches to TRUE.
ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

bReadingBrightnessN: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevelN: Measured brightness value of the light sensor.

These measured values must be compared with the measured reference values at the place of operation of the DALI control device.

nPushButtonEventN: The output outputs the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightnessN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightnessN.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancyN.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input that can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBrightnessN</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nRoomCorrectionFactorN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30</td>
<td>The room correction factor (5…200) specifies the difference in brightness measurement on the ceiling and at the workplace. It can be calculated using a luxmeter with the following equation: $RoomCorrectionFactor = \frac{Brightness \text{ under ceiling}}{Brightness \text{ on work surface}}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The value can also be determined by a remote control and with an appropriate app. The preset value of 0.3 (30) is suitable for most applications.</td>
</tr>
<tr>
<td>nReportTimerBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimerBrightnessN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>163</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E.DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>
### Name | Type | Access | Initial value | Description
---|---|---|---|---
nSensitivityOcc | BYTE | Get, Set | 3 | Setting of the sensitivity of both occupancy sensor instances from 0 (low sensitivity) to 5 (high sensitivity).
eEventPriorityPushButtonN | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.11.4 FB_DALI_Theben_theRonda_S360

```c
void FB_DALI_Theben_theRonda_S360(unsigned int instanceNumber)
```

The function block represents the theRonda S360 DALI sensor from Theben.

This outputs an occupancy signal (instance 0), a measured brightness value (instance 1) and eight push button signals (instances 5 to 12).

The push button instances can be used via an infrared remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

- Part 301 (push buttons) - [Introduction](#)
- Part 303 (occupancy sensors) – [Introduction](#)
- Part 304 (light sensors) – [Introduction](#)
The existing parameters can be overwritten by changing the properties listed below.

Readout can be triggered manually via the input variables `bQueryOccupancy` and `bQueryBrightness`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
<td>The occupancy is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
<td>The brightness is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL := FALSE;</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output <code>bOccupied</code>.</td>
</tr>
</tbody>
</table>

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable <code>ipResultMessage</code>. The output is set to FALSE again as soon as <code>bBusy</code> switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after <code>bBusy</code> has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bInitializing</td>
<td>The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.</td>
</tr>
</tbody>
</table>
bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

nPushButtonEventN: The output outputs the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input $bQueryBrightness$.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input $bQueryOccupancy$.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input that can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALI</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority $[160]$.</td>
</tr>
<tr>
<td>nRoomCorrectionFactor</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30</td>
<td>The room correction factor (5…200) specifies the difference in brightness measurement on the ceiling and at the workplace. It can be calculated using a luxmeter with the following equation: $\text{RoomCorrectionFactor} = \frac{\text{Brightness under ceiling}}{\text{Brightness on work surface}}$.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The value can also be determined by a remote control and with an appropriate app. The preset value of 0.3 (30) is suitable for most applications.</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimeBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightnessN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>163</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALI</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority $[160]$.</td>
</tr>
<tr>
<td>nDeadtimeTimeOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensitivityOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>3</td>
<td>Setting of the sensitivity of both occupancy sensor instances from 0 (low sensitivity) to 5 (high sensitivity).</td>
</tr>
</tbody>
</table>
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.11.5 FB_DALI_Theben_TA_4_S

The function block represents the TA 4 S DALI-2 DALI push-button interface from Theben.

Up to 4 push buttons (instances 2 to 5) and 2 slider/temperature sensors (instances 0 and 1) can be connected via the device.

The push button inputs are configured as make contact (NO) in the delivery state. You can change the contact type to break contact (NC) via the properties (see Properties [510]).

You can set the parameters of the two absolute inputs via the properties (see Properties [510]). When delivered, 100 kOhm variable resistors can be connected directly.

For further information on configuration, please refer to the manufacturer’s device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) – Introduction [288]

Part 302 (absolute encoder) - Introduction [304]

All inputs can be queried immediately via a positive edge at `bQueryInputValueN` and `bQueryAbsoluteInputN`. 

---

### Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DALI</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100…2000 ms) of repetitive events in the case of a long keystroke. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.</td>
</tr>
</tbody>
</table>
The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at \textit{bInitialize}.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

\textbf{VAR_INPUT}

\begin{verbatim}
VAR_INPUT

bInitialize        : BOOL := FALSE;
nShortAddress      : BYTE := 0;
nOptions           : DWORD := 0;
bQueryInputValue1  : BOOL := FALSE;
bQueryInputValue2  : BOOL := FALSE;
bQueryInputValue3  : BOOL := FALSE;
bQueryInputValue4  : BOOL := FALSE;
bQueryAbsoluteInput1: BOOL := FALSE;
bQueryAbsoluteInput2: BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryInputValueN: Direct query of the values at the button inputs 1 to 4.

bQueryAbsoluteInputN: Immediate query of the values at the absolute value inputs 1 to 2.
\end{verbatim}

\textbf{VAR_OUTPUT}

\begin{verbatim}
VAR_OUTPUT

bError            : BOOL;
ipResultMessage  : I_TcMessage;
bInitializing    : BOOL;
bReadingInputValue1 : BOOL;
bReadingInputValue2 : BOOL;
bReadingInputValue3 : BOOL;
bReadingInputValue4 : BOOL;
bReadingAbsoluteInputValue1 : BOOL;
bReadingAbsoluteInputValue2 : BOOL;
nPushButton1Event : WORD;
nPushButton2Event : WORD;
nPushButton3Event : WORD;
nPushButton4Event : WORD;
nAbsoluteInput1   : BYTE;
nAbsoluteInput2   : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable \textit{ipResultMessage}. The output is set to FALSE again as soon as \textit{bBusy} switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation \[574\]) that can be used to obtain detailed information about the processing of the function block (see runtime messages \[561\]). The interface pointer is valid after \textit{bBusy} has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.
\end{verbatim}
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block `FB_DALI_ToPushButtonState` can be used to convert the state of a push button into a BOOL variable.

`nAbsoluteInputN`: Outputs containing the measured values, [°C] or [Ω], of the connected variable resistors. Please refer to the manufacturer documentation of the TA 4 S DALI-2 for the appropriate scaling of the output value.

**Properties**

All parameters that are written to the DALI control device via `bInitialize` are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>Push button input 1 to 4, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.</td>
</tr>
<tr>
<td>nEventFilterPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter ([288].)</td>
</tr>
<tr>
<td>eEventPriorityInputN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority ([160].)</td>
</tr>
<tr>
<td>nDeadtimeTimerAbsoluteInputN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
<tr>
<td>nStuckTimerPushButtonN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>20 s</td>
<td>Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.</td>
</tr>
<tr>
<td>bEnableAbsoluteInputN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Absolute value input 1 or 2, which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryAbsoluteInputN.</td>
</tr>
<tr>
<td>eEventPriorityAbsoluteInputN</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority ([160].)</td>
</tr>
<tr>
<td>nDeadtimeTimerAbsoluteInputN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nRepeatTimerAbsoluteInputN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>0 s</td>
<td>Time [s] after which the pending event of the absolute encode is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nSensorTypeAbsoluteInputN</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>16#03</td>
<td>Sensor type: 16#00: Theben sensors (9070321, 9070459, 9070489, 9070496). 16#01: Theben sensor (9070191). 16#02: any 100 kΩ NTC with known B or Beta [K] (see also value range parameter Beta Low / Beta High below). 16#03: 100 kΩ variable resistor (delivery state).</td>
</tr>
</tbody>
</table>
**Name** | **Type** | **Access** | **Initial value** | **Description**
---|---|---|---|---
nBetaHighAbsoluteInputN | BYTE | Get, Set | 16#11 | If sensor type 16#02 has been selected, the associated B or Beta [K] should be set here. The B or Beta is a four-digit number in [K], which can be found in the data sheet of the NTC sensor. For storage in the memory bank, the four-digit number must first be converted to hex format and then stored in the Beta Low and Beta High bytes. Example: B or Beta = 4580, this corresponds to 16#11E4 • Beta Low = 16#E4, delivery state • Beta High = 16#11, delivery state • The setting range is 0x00...0xFF.

nBetaLowAbsoluteInputN | BYTE | Get, Set | 16#E4 | See nBetaHighAbsoluteInputN

nFilterAbsoluteInputN | BYTE | Get, Set | 16#00 | A filter can be set for noise suppression. The setting range is 16#00...16#OB. 16#00: each measured value is transmitted unchanged (delivery status) 16#01: average of 2 measured values (2¹). 16#02: average of 4 measured values (2²). 16#00: average of 8 measured values (2³). ... 16#00: average of 512 measured values (2⁹). 16#00: average of 1024 measured values (2¹⁰). 16#00: average of 2048 measured values (2¹¹). **Attention**: The filter delays the measurement!

nHysteresisAbsoluteInputN | BYTE | Get, Set | 16#00 | Hysteresis value to avoid frequent events. The setting range is 16#00...16#FF. This corresponds to a value range of 0.1 °C...25.5 °C for sensor types 16#00...16#02 or a value range of 100 Ω...25.5 kΩ for sensor type 16#03. The step size of the hysteresis corresponds to 0.1 °C or 100 Ω.

nOffsetAbsoluteInputN | BYTE | Get, Set | 16#80 | The setting range of the offset is 16#01...16#FF. This corresponds to a value range of -12.7 °C...+12.7 °C for sensor types 16#00...16#02 or a value range of -12.7 kΩ...+12.7 kΩ for sensor type 16#03. The step size of the offset corresponds to 0.1 °C or 0 Ω. The 16#80 in the delivery state corresponds to 0 °C or 0 Ω.

nRangeAbsoluteInputN | BYTE | Get, Set | 16#00 | Setting of the Range parameter: 16#00: Temperature sensors with a positive temperature range (delivery state). 16#01: Temperature sensors with a negative temperature range.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.20</td>
<td>Tc3_DALI from v3.7.0.0</td>
</tr>
</tbody>
</table>
3.1.4.12  Tridonic

3.1.4.12.1  FB_DALI_Tridonic_MSensorG3_SFI_30_PIR

The function block represents the MSensor G3 SFI 30 PIR 5DPI, MSensor G3 SFI 30 PIR 10DPI or MSensor G3 SFI 30 PIR 16DPI DALI sensor from Tridonic.

This outputs an occupancy signal (instance 0), a measured brightness value (instance 1) and six push button signals (instances 2 to 7).

The push button instances can be used via an infrared remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]
Part 303 (occupancy sensors) – Introduction [313]
Part 304 (light sensors) – Introduction [327]

The existing parameters can be overwritten by changing the properties listed below.

The input variables `bQueryOccupancy` and `bQueryBrightness` trigger the immediate readout of the corresponding instance.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE := 0;</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD := 0;</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL := FALSE;</td>
</tr>
<tr>
<td>bCancelHoldtimerOccupancy</td>
<td>BOOL := FALSE;</td>
</tr>
</tbody>
</table>

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.
**nOptions:** Reserved for future extensions.

**bQueryOccupancy:** The occupancy is queried immediately by a positive edge at this input.

**bQueryBrightness:** The brightness is queried immediately by a positive edge at this input.

**bCancelHoldTimerOccupancy:** A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT</td>
</tr>
<tr>
<td>nPushButton1Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton2Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton3Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton4Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton5Event</td>
<td>WORD</td>
</tr>
<tr>
<td>nPushButton6Event</td>
<td>WORD</td>
</tr>
</tbody>
</table>

**bError:** This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

**ipResultMessage:** Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

**bInitializing:** The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

**bQueryOccupancy:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bQueryBrightness:** The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

**bOccupied:** If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

**nBrightnessLevel:** Measured brightness value of the light sensor.

> This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

**nPushButtonNEvent:** The output outputs the events of the push buttons via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.
PROPERTIES

All parameters that are written to the DALI control device via \textit{bInitailize} are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Name | Type | Access | Initial Value | Description
---|---|---|---|---
bEnableBrightness | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.
bEnableOccupancy | BOOL | Get, Set | TRUE | TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.
bEnablePushButtonN | BOOL | Get, Set | FALSE | Push button input that can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.
eEventPriorityBrightness | E_DALIEventPriority | Get, Set | Middle Low | Setting of the event priority [160].
nReportTimerBrightness | BYTE | Get, Set | 30 s | Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.
nDeadtimeTimeBrightness | UINT | Get, Set | 1500 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
hysteresisBrightness | BYTE | Get, Set | 5 % | Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.
hysteresisMinBrightness | BYTE | Get, Set | 255 | As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.
eEventPriorityOccupancy | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
nDeadtimeTimeOccupancy | UINT | Get, Set | 100 ms | Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.
holdTimerOccupancy | UINT | Get, Set | 900 s | Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.
nReportTimerOccupancy | BYTE | Get, Set | 60 s | Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.
eEventPriorityPushButtonN | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].
EVENTFILTER | BYTE | Get, Set | 2#1000_0011 | Setting of the event filter [288].
DoubleTimerPushButtonN | UINT | Get, Set | 0 ms | Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.
RepeatTimerPushButtonN | UINT | Get, Set | 160 ms | Interval (100…2000 ms) of repetitive events in the case of a long keystroke. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.
ShortTimerPushButtonN | UINT | Get, Set | 500 ms | Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.12.2 FB_DALI_Tridonic_MSensorG3_SRC_30_PIR

The function block represents the MSensor G3 SRC 30 PIR 5DPI DALI sensor from Tridonic.

This outputs an occupancy signal (instance 0), a measured brightness value (instance 1) and six push button signals (instances 2 to 7).

The push button instances can be used via an infrared remote control from the manufacturer.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

- Part 301 (push buttons) - [Introduction](#) [288]
- Part 303 (occupancy sensors) – [Introduction](#) [313]
- Part 304 (light sensors) – [Introduction](#) [327]

The existing parameters can be overwritten by changing the properties listed below.

The input variables `bQueryOccupancy` and `bQueryBrightness` trigger the immediate readout of the corresponding instance.
VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryOccupancy : BOOL := FALSE;
bQueryBrightness : BOOL := FALSE;
bCancelHoldTimerOccupancy : BOOL := FALSE;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryOccupancy: The occupancy is queried immediately by a positive edge at this input.

bQueryBrightness: The brightness is queried immediately by a positive edge at this input.

bCancelHoldTimerOccupancy: A positive edge at this input prematurely terminates the Hold Timer and resets the output bOccupied.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingOccupancy : BOOL;
bReadingBrightness : BOOL;
bOccupied : BOOL;
nBrightnessLevel : UINT;
nPushButton1Event : WORD;
nPushButton2Event : WORD;
nPushButton3Event : WORD;
nPushButton4Event : WORD;
nPushButton5Event : WORD;
nPushButton6Event : WORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingOccupancy: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bReadingBrightness: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

nPushButtonNEvent: The output outputs the events of the push buttons via one bit each for a cycle.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via blInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryBrightness.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryOccupancy.</td>
</tr>
<tr>
<td>bEnablePushButtonN</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>Push button input that can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadtimeTimeBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>255</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimeOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>eEventPriorityPushButtonN</td>
<td>E_DALIEEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nEventFilterPushButtonInternal</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#1000_0011</td>
<td>Setting of the event filter [288].</td>
</tr>
<tr>
<td>nDoubleTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>0 ms</td>
<td>Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.</td>
</tr>
<tr>
<td>nRepeatTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>160 ms</td>
<td>Interval (100…2000 ms) of repetitive events in the case of a long keystroke. This property does not apply to the instances PushButtonScene1 and PushButtonScene2.</td>
</tr>
<tr>
<td>nShortTimerPushButtonN</td>
<td>UINT</td>
<td>Get, Set</td>
<td>500 ms</td>
<td>Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.</td>
</tr>
</tbody>
</table>
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.12.3 FB_DALI_Trionic_XC_G3_CWM30

The function block represents the DALI XC G3 CWM 30 DA2 DALI push-button interface from Tridonic.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]

The existing parameters can be overwritten by changing the properties listed below.

The input variables `bQueryInputValueN` trigger the manual reading of the corresponding instance.

The properties `bEnablePushButtonN` make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at `bInitialize` and stored there.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

### VAR_INPUT

```plaintext
VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nOptions : DWORD := 0;
bQueryInputValue1 : BOOL := FALSE;
bQueryInputValue2 : BOOL := FALSE;
bQueryInputValue3 : BOOL := FALSE;
bQueryInputValue4 : BOOL := FALSE;
```
bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryInputValueN: The state of the corresponding push button is queried immediately by a positive edge at this input.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage;</td>
</tr>
<tr>
<td>bInitialize</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue1</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue2</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue3</td>
<td>BOOL;</td>
</tr>
<tr>
<td>bReadingInputValue4</td>
<td>BOOL;</td>
</tr>
<tr>
<td>nPushButton1Event</td>
<td>WORD;</td>
</tr>
<tr>
<td>nPushButton2Event</td>
<td>WORD;</td>
</tr>
<tr>
<td>nPushButton3Event</td>
<td>WORD;</td>
</tr>
<tr>
<td>nPushButton4Event</td>
<td>WORD;</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

**PROPERTIES**

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
bEnablePushButtonN | BOOL | Get, Set | TRUE | Push button input 1...4, which can be either enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButtonN.
nEventFilterPushButtonN | BYTE | Get, Set | 2#100001 | Setting of the event filter [p 288].
eEventPriorityPushButtonN | E.DALIEventPriority | Get, Set | Middle | Setting of the event priority [p 160].
nDoubleTimerPushButtonN | UINT | Get, Set | 0 ms | Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.
nRepeatTimerPushButtonN | UINT | Get, Set | 160 ms | Interval (100…2000 ms) of repetitive events in the case of a long keystroke.
nShortTimerPushButtonN | UINT | Get, Set | 500 ms | Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.
nStuckTimerPushButtonN | BYTE | Get, Set | 20 s | Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.

### Requirements

**Development environment**

<table>
<thead>
<tr>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
</tr>
</tbody>
</table>

#### 3.1.4.13 Zencontrol

##### 3.1.4.13.1 FB_DALI_Zencontrol_PIR

The function block represents the PIR DALI-2 sensor from Zencontrol.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – [Introduction p 313]

Part 304 (light sensors) – [Introduction p 327]

Readout can be triggered immediately via the input variables bQueryOccupancy and bQueryBrightness.

The existing parameters can be overwritten by changing the properties listed below. The parameters are written to the device after a positive edge at bInitialize.
Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
</tr>
<tr>
<td>bQueryOccupancy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bQueryBrightness</td>
<td>BOOL</td>
</tr>
<tr>
<td>bCancelHoldTimerOccupancy</td>
<td>BOOL</td>
</tr>
</tbody>
</table>

*bInitialize*: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

*nShortAddress*: Short address (0…63) of the DALI control device.

*nOptions*: Reserved for future extensions.

*bQueryOccupancy*: The occupancy is queried immediately by a positive edge at this input.

*bQueryBrightness*: The brightness is queried immediately by a positive edge at this input.

*bCancelHoldTimerOccupancy*: A positive edge at this input prematurely terminates the Hold Timer and resets the output *bOccupied*.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingOccupancy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingBrightness</td>
<td>BOOL</td>
</tr>
<tr>
<td>bOccupied</td>
<td>BOOL</td>
</tr>
<tr>
<td>nBrightnessLevel</td>
<td>UINT</td>
</tr>
</tbody>
</table>

*bError*: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable *ipResultMessage*. The output is set to FALSE again as soon as *bBusy* switches to TRUE.

*ipResultMessage*: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after *bBusy* has changed from TRUE to FALSE.

*bInitializing*: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

*bReadingOccupancy*: The output is TRUE as long as values of the instance of the motion sensor are being read by the DALI control device.

*bReadingBrightness*: The output is TRUE as long as values of the instance of the light sensor are being read by the DALI control device.

*bOccupied*: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

*nBrightnessLevel*: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.
PROPERTIES

All parameters that are written to the DALI control device via \textit{bInitialize} are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableBrightness</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for brightness measurement. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \textit{bQueryBrightness}.</td>
</tr>
<tr>
<td>bEnableOccupancy</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>TRUE at this property enables the instance for motion detection. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input \textit{bQueryOccupancy}.</td>
</tr>
<tr>
<td>eEventPriorityBrightness</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nReportTimerBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
<tr>
<td>nDeadlineTimerBrightness</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresisBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the brightness.</td>
</tr>
<tr>
<td>nHysteresisMinBrightness</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>As the percentage hysteresis can also lead to large fluctuations at low brightness, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255.</td>
</tr>
<tr>
<td>eEventPriorityOccupancy</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadlineTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimerOccupancy</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1…2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimerOccupancy</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0…255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.9.0.0</td>
</tr>
</tbody>
</table>
3.1.4.13.2 FB_DALI_Zencontrol_Scenepanel_Switch

The function block represents the Scenepanel Switch DALI push-button interface from Zencontrol.

Up to four push buttons (instances 0 to 3) can be connected directly via the device.

For information on the exact functions of the individual instances, please refer to the manufacturer's device documentation.

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]

The existing parameters can be overwritten by changing the properties listed below.

The input variables bQueryInputValueN trigger the immediate reading of the corresponding instance.

The properties bEnablePushButtonN make it possible to block individual instances.

All parameters are written to the DALI control device by a positive edge at bInitialize and stored there.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>bQueryInputValue1</td>
<td>BOOL</td>
<td>The state of the corresponding push button is queried immediately by a positive edge at this input.</td>
</tr>
<tr>
<td>bQueryInputValue2</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bQueryInputValue3</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>bQueryInputValue4</td>
<td>BOOL</td>
<td></td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
</tbody>
</table>
bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValueN: The outputs are set as soon as the manual reading of the corresponding input has been triggered.

nPushButtonNEvent: The outputs output the events for the respective push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

The properties of instances of the same type are listed only once in the table and marked with N at the end.
### Name | Type | Access | Initial value | Description
--- | --- | --- | --- | ---
'bEnablePushButtonN' | BOOL | Get, Set | TRUE | Push button input 1...4, which can be either enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input 'bQueryPushButtonN'.

'nEventFilterPushButtonN' | BYTE | Get, Set | 2#100 0_001 | Setting of the event filter [288].

'eEventPriorityPushButtonN' | E.DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].

'nDoubleTimerPushButtonN' | UINT | Get, Set | 0 ms | Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.

'nRepeatTimerPushButtonN' | UINT | Get, Set | 160 ms | Interval (100...2000 ms) of repetitive events in the case of a long keystroke.

'nShortTimerPushButtonN' | UINT | Get, Set | 500 ms | Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.

'nStuckTimerPushButtonN' | BYTE | Get, Set | 20 s | Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.9.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.13.3 FB_DALI_Zencontrol_Switch_1Button

The function block represents the Switch 1 Button DALI push-button interface from Zencontrol.

A push button (instance 0) can be connected directly via the device.

For information on the exact functions of the instance, please refer to the manufacturer’s device documentation.

Further information on the supported instance type can be found here:

Part 301 (push buttons) - Introduction [288]

The existing parameters can be overwritten by changing the properties listed below.

The input variable 'bQueryInputValue' triggers the immediate reading of the instance.

It is possible to block the instance via the property 'bEnablePushButton'.

All parameters are written to the DALI control device by a positive edge at 'bInitialize' and stored there.

---

**Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.**
The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>0</td>
</tr>
<tr>
<td>bQueryInputValue</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nOptions: Reserved for future extensions.

bQueryInputValue: The state of the corresponding push button is queried immediately by a positive edge at this input.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue</td>
<td>BOOL</td>
</tr>
<tr>
<td>nPushButtonEvent</td>
<td>WORD</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingValue: The output is set as soon as the manual reading of the input has been triggered.

nPushButtonEvent: The output outputs the events for the push button via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Name | Type | Access | Initial Value | Description
--- | --- | --- | --- | ---
**bEnablePushButton** | BOOL | Get, Set | TRUE | Push button input which can be enabled or disabled on the DALI control device. FALSE at this property causes sending of events from this instance to be disabled. However, the actual value of the instance can still be queried via the input bQueryPushButton.

**nEventFilterPushButton** | BYTE | Get, Set | 2#100 0_001 | Setting of the event filter [288].

**eEventPriorityPushButton** | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority [160].

**nDoubleTimerPushButton** | UINT | Get, Set | 0 ms | Time (0, tDoubleMin…2000 ms) to distinguish between a single and a double keystroke.

**nRepeatTimerPushButton** | UINT | Get, Set | 160 ms | Interval (100…2000 ms) of repetitive events in the case of a long keystroke.

**nShortTimerPushButton** | UINT | Get, Set | 500 ms | Time (tShortMin…5100 ms) to distinguish between a long and a short keystroke. tShortMin is a manufacturer-specific value.

**nStuckTimerPushButton** | BYTE | Get, Set | 20 s | Time (5…255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.

### Requirements

**Development environment**

<table>
<thead>
<tr>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
</tr>
</tbody>
</table>

### 3.1.4.14 **FB_DALI103ControlDevice**

The function block is used for the configuration and operation of a DALI control unit that supports Part 103 (control units).

The application controller can be activated, the operation mode specified and the event **Power Cycle Notification** enabled.

**VAR_INPUT**

```plaintext
bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
bGetPowerCycleNotifications : BOOL := TRUE;
nOptions : DWORD := 0;
```

**bInitialize**: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

**nShortAddress**: Short address (0…63) of the DALI control device.

**bGetPowerCycleNotifications**: A TRUE at this input causes the received **Power Cycle Notification** to be output at the output.
**nOptions:** Reserved for future extensions.

### VAR OUTPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
<td></td>
<td></td>
<td>Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
<td></td>
<td>FALSE</td>
<td>The output is set as soon as the initialization of the DALI device has been started, and remains active until all DALI commands have been executed.</td>
</tr>
<tr>
<td>bPowerCycleNotification</td>
<td>BOOL</td>
<td></td>
<td>FALSE</td>
<td>As soon as a Power Cycle Notification Event matches the desired filter criteria, this output is set to TRUE for one PLC cycle.</td>
</tr>
<tr>
<td>nPowerCycleNotificationShortAddress</td>
<td>BYTE</td>
<td></td>
<td>0</td>
<td>This output contains the short address (0…63) of the DALI control device that sent the Power Cycle Notification Event.</td>
</tr>
</tbody>
</table>

### PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableApplicationController</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>If this property is TRUE, the application controller (see Device variables [157]) is activated.</td>
</tr>
<tr>
<td>bEnablePowerCycleNotification</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>FALSE</td>
<td>TRUE at this property enables the Power Cycle Notification event.</td>
</tr>
<tr>
<td>nOperatingMode</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>0</td>
<td>Specification of the operation mode (0…255) of the DALI control device (see Device variables [157]).</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

**3.1.4.15 FB_DALI300GenericInstance**

The function block is used for the direct reading of generic values of an instance of a DALI control device that supports Part 103 (instance type 0).

The input variable bEnableInstance can be used to configure whether the instance is enabled.

If necessary, the output value can also be queried immediately using the variable bQueryInputValue.
The function block cannot be used when using the KL6811.

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>nInstanceNumber</td>
<td>BYTE</td>
<td>0</td>
</tr>
<tr>
<td>nResolution</td>
<td>BYTE</td>
<td>10</td>
</tr>
<tr>
<td>bQueryInputValue</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>0</td>
</tr>
</tbody>
</table>

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nInstanceNumber: Number of the instance (0…31) of the DALI control device to be addressed.

nResolution: Manufacturer-specific resolution (1…64) of the input value. The value specifies the number of bits with which the input value is scaled.

bQueryInputValue: Immediate query of the measured value. A query is also possible if the instance is locked via the property bEnableInstance.

nOptions: Reserved for future extensions.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>ipResultMessage</td>
<td>I_TcMessage</td>
</tr>
<tr>
<td>bInitializing</td>
<td>BOOL</td>
</tr>
<tr>
<td>bReadingInputValue</td>
<td>BOOL</td>
</tr>
<tr>
<td>nValue</td>
<td>ULINT</td>
</tr>
</tbody>
</table>

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValue: The output is set as soon as the manual reading of the measured value has been started and remains active until all DALI commands have been processed.

nValue: Outputs the measured value.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.16 FB_DALI301PushButton

The function block is used for the configuration and operation of an instance of a DALI control device that supports Part 301 (push buttons).

Further information on the supported instance types can be found here:

Part 301 (push buttons) - Introduction [288]

If necessary, the output value can also be queried immediately using the variable `bQueryInputValue`.

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

- The function block cannot be used when using the KL6811.

### VAR_INPUT

- `bInitial`e: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

- `nShortAddress`: Short address (0…63) of the DALI control device.

- `nInstanceNumber`: Number of the instance (0…31) of the DALI control device to be addressed.

- `bQueryInputValue`: The state of the input is queried immediately by a positive edge at this input. A query is also possible if the instance is locked via the property `bEnableInstance`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableInstance</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>The instance can be enabled or disabled on the DALI control device. If this property is FALSE, sending events from this instance is disabled. However, the actual value of the instance can still be queried via the input <code>bQueryInputValue</code>.</td>
</tr>
<tr>
<td>nEventFilter</td>
<td>DWORD</td>
<td>Get, Set</td>
<td>16#00_00_00_00_00</td>
<td>Setting of the event filter.</td>
</tr>
<tr>
<td>eEventPriority</td>
<td>E_DALI  EventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the <code>event_priority</code> [160].</td>
</tr>
</tbody>
</table>
bGetInputNotifications: A TRUE at this input causes the received Input Notification Events to be output at the output.

nOptions: Reserved for future extensions.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingInputValue : BOOL;
nPushButtonEvent : WORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValue: The output is set as soon as the manual reading of the measured value has been started and remains active until all DALI commands have been processed.

nPushButtonEvent: The output outputs the events via one bit each for a cycle.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block FB_DALI_ToPushButtonState [542] can be used to convert the state of a push button into a BOOL variable.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Name | Type | Access | Initial value | Description
---|---|---|---|---
bEnableInstance | BOOL | Get, Set | TRUE | The instance can be enabled or disabled on the DALI control device. If this property is FALSE, sending events from this instance is disabled. However, the actual value of the instance can still be queried via the input bQueryInputValue.
nEventFilter | BYTE | Get, Set | 2#1000_00_11 | Setting of the event filter `[288]`.
eEventPriority | E_DALIEventPriority | Get, Set | Middle | Setting of the event priority `[160]`.
nRepeatTime | UINT | Get, Set | 160 ms | Time (100...2000 ms) after which the pending event of the push button is repeated in case of a long keystroke.
nShortTimer | UINT | Get, Set | 500 ms | Time (tShortMin...5100 ms) to distinguish between a long and a short keystroke.
nDoubleTimer | UINT | Get, Set | 0 ms | Time (0, tDoubleMin...2000 ms) to distinguish between a single and a double keystroke.
nStuckTimer | BYTE | Get, Set | 20 s | Time (5...255 s) after the expiry of which the push button no longer represents a long keystroke. The push button is stuck.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.1.4.17 FB_DALI302AbsoluteInputDevice

The function block is used for the configuration and operation of an instance of a DALI control device that supports Part 302 (absolute input device).

The Properties `[536]` can be used to configure whether the instance is enabled and how it behaves with respect to time (timer inputs). In addition, values can also be queried manually if necessary via the variable bQueryInputValue.

The output variable nInputLevel is written depending on the specified resolution nResolution (application and vendor-specific).

- Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.
- The function block cannot be used when using the KL6811.
VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nInstanceNumber : BYTE := 0;
nResolution : BYTE := 1;
bQueryInputValue : BOOL := FALSE;
bGetInputNotifications : BOOL := TRUE;
nOptions : DWORD := 0;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nInstanceNumber: Number of the instance (0…31) of the DALI control device to be addressed.

nResolution: Manufacturer-specific resolution (1…64) of the input value. The value specifies the number of bits with which the input value is scaled.

bQueryInputValue: Immediate query of the measured value. A query is also possible if the instance is locked via the property bEnableInstance.

bGetInputNotifications: A TRUE at this input causes the received Input Notification Events to be output at the output.

nOptions: Reserved for future extensions.

VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingInputValue : BOOL;
nInputLevel : BYTE;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValue: The output is set as soon as the manual reading of the measured value has been started and remains active until all DALI commands have been processed.

nInputLevel: Outputs the measured value of the resistance measurement.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.
### Name: bEnableInstance
**Type:** BOOL  | **Access:** Get, Set  | **Initial value:** TRUE
**Description:** The instance can be enabled or disabled on the DALI control device. If this property is FALSE, sending events from this instance is disabled. However, the actual value of the instance can still be queried via the input bQueryInputValue.

### Name: eEventPriority
**Type:** E.DALIEventPriority  | **Access:** Get, Set  | **Initial value:** Middle
**Description:** Setting of the event priority [160].

### Name: nReportTimer
**Type:** BYTE  | **Access:** Get, Set  | **Initial value:** 0
**Description:** Time (0…255 s) after which the pending event of the measurement is repeated if no other event has occurred in the meantime.

### Name: nDeadtimeTimer
**Type:** UINT  | **Access:** Get, Set  | **Initial value:** 100 ms
**Description:** Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.

### Examples of the setting of the resolution:

#### Simple switch (make contact)

```plaintext
nResolution := 1;
```

When the contact is open, the output has the following value:

```plaintext
nInputLevel := 0;
```

When closed, the output has the following value:

```plaintext
nInputLevel := 1;
```

#### Switch with two positions

```plaintext
nResolution := 2;
```

When contact 1 is closed, the output has the following value:

```plaintext
nInputLevel := 1;
```

When contact 2 is closed, the output has the following value:

```plaintext
nInputLevel := 2;
```

#### Switch with two positions and a neutral center position:

```plaintext
nResolution := 2;
```

When contact 1 is closed, the output has the following value:

```plaintext
nInputLevel := 1;
```

In the center position, the output has the following value:

```plaintext
nInputLevel := 0
```

When contact 2 is closed, the output has the following value:

```plaintext
nInputLevel := 2;
```

#### Rotary switch:

When using a rotary switch with latching positions, the resolution `nResolution` depends on the number of available positions.

The output of the value `nInputLevel` also corresponds to this.

#### Slide resistor, potentiometer (absolute input device):

When using an absolute input device, the resolution `nResolution` and the output of the variable `nInputLevel` are vendor-dependent.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

### 3.1.4.18 FB_DALI303OccupancySensor

The function block is used for the configuration and operation of an instance of a DALI control device that supports Part 303 (occupancy sensor).

Further information on the supported instance types can be found here:

Part 303 (occupancy sensors) – [Introduction](#)

The properties can be used to configure whether the instance is enabled and how it behaves with regard to time (timer). In addition, values can also be queried immediately if necessary via the input variable `bQueryInputValue`.

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

### VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bInitialize</td>
<td>BOOL</td>
<td>The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.</td>
</tr>
<tr>
<td>nShortAddress</td>
<td>BYTE</td>
<td>Short address (0…63) of the DALI control device.</td>
</tr>
<tr>
<td>nInstanceNumber</td>
<td>BYTE</td>
<td>Number of the instance (0…31) of the DALI control device to be addressed.</td>
</tr>
<tr>
<td>bQueryInputValue</td>
<td>BOOL</td>
<td>Immediate query of the measured value. A query is also possible if the instance is locked via the property <code>bEnableInstance</code>.</td>
</tr>
<tr>
<td>bGetInputNotifications</td>
<td>BOOL</td>
<td>A TRUE at this input causes the received Input Notification Events to be output at the output.</td>
</tr>
<tr>
<td>bCancelHoldTimer</td>
<td>BOOL</td>
<td>A positive edge at this input prematurely terminates the Hold Timer and resets the output <code>bOccupied</code>.</td>
</tr>
<tr>
<td>nOptions</td>
<td>DWORD</td>
<td>Reserved for future extensions.</td>
</tr>
</tbody>
</table>
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingInputValue : BOOL;
bOccupied : BOOL;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValue: The output is set as soon as the manual reading of the measured value has been started and remains active until all DALI commands have been processed.

bOccupied: If the output is TRUE, then occupancy has been detected in the detection range of the occupancy sensor.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableInstance</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>The instance can be enabled or disabled on the DALI control device. If this property is FALSE, sending events from this instance is disabled. However, the actual value of the instance can still be queried via the input bQueryInputValue.</td>
</tr>
<tr>
<td>nEventFilter</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>2#0000_0111</td>
<td>Setting of the event filter [314].</td>
</tr>
<tr>
<td>eEventPriority</td>
<td>E_DALIEventPriority</td>
<td>Get, Set</td>
<td>Middle</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimer</td>
<td>UINT</td>
<td>Get, Set</td>
<td>100 ms</td>
<td>Dead time (0...12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHoldTimer</td>
<td>UINT</td>
<td>Get, Set</td>
<td>900 s</td>
<td>Dwell time (1...2538 s) in which the occupancy continues to be active after no more movement has been detected by the sensor.</td>
</tr>
<tr>
<td>nReportTimer</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>60 s</td>
<td>Time (0...255 s) after which the pending event of the occupancy sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
3.1.4.19 FB_DALI304LightSensor

The function block is used for the configuration and operation of an instance of a DALI control device that supports Part 304 (light sensor).

Further information on the supported instance types can be found here:

Part 304 (light sensor) - Introduction [327]

The properties can be used to configure whether the instance is enabled and how it behaves with regard to time (timer). In addition, values can also be queried immediately if necessary via the input variable bQueryInputValue.

The output variable nBrightnessLevel is written depending on the specified resolution nResolution (application and manufacturer-specific).

Set the times so that as few events as possible are sent. If too many events are sent, this can have a negative effect on the behavior of the application.

The function block cannot be used when using the KL6811.

VAR_INPUT

bInitialize : BOOL := FALSE;
nShortAddress : BYTE := 0;
nInstanceNumber : BYTE := 0;
nResolution : BYTE := 10;
bQueryInputValue : BOOL := FALSE;
bGetInputNotifications : BOOL := TRUE;
nOptions : DWORD := 0;

bInitialize: The configuration of the DALI control device is started by a positive edge at this input. During this time no DALI commands are processed.

nShortAddress: Short address (0…63) of the DALI control device.

nInstanceNumber: Number of the instance (0…31) of the DALI control device to be addressed.

nResolution: Manufacturer-specific resolution (1…64) of the input value. The value specifies the number of bits with which the input value is scaled.

bQueryInputValue: Immediate query of the measured value. A query is also possible if the instance is locked via the property bEnableInstance.

bGetInputNotifications: A TRUE at this input causes the received Input Notification Events to be output at the output.

nOptions: Reserved for future extensions.
VAR_OUTPUT

bError : BOOL;
ipResultMessage : I_TcMessage;
bInitializing : BOOL;
bReadingInputValue : BOOL;
nBrightnessLevel : ULINT;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bInitializing: The output is set as soon as the initialization of the DALI control device has been started, and remains active until all DALI commands have been executed.

bReadingInputValue: The output is set as soon as the manual reading of the measured value has been started and remains active until all DALI commands have been processed.

nBrightnessLevel: Measured brightness value of the light sensor.

This measured value must be compared with the measured reference values at the place of operation of the DALI control device.

PROPERTIES

All parameters that are written to the DALI control device via bInitialize are available as properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Initial value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnableInstance</td>
<td>BOOL</td>
<td>Get, Set</td>
<td>TRUE</td>
<td>The instance can be enabled or disabled on the DALI control device. If this property is FALSE, sending events from this instance is disabled. However, the actual value of the instance can still be queried via the input bQueryInputValue.</td>
</tr>
<tr>
<td>eEventPriority</td>
<td>E_DALI_EventPriority</td>
<td>Get, Set</td>
<td>MiddleLow</td>
<td>Setting of the event priority [160].</td>
</tr>
<tr>
<td>nDeadtimeTimer</td>
<td>UINT</td>
<td>Get, Set</td>
<td>1500 ms</td>
<td>Dead time (0…12750 ms) to ensure that not too many events are sent in succession. An event is not sent until the dead time timer has expired. The dead time timer is restarted each time an event is sent.</td>
</tr>
<tr>
<td>nHysteresis</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>5 %</td>
<td>Value (0…25 %) for calculating a hysteresis value in order to prevent frequent and disturbing changes when measuring the illuminance.</td>
</tr>
<tr>
<td>nHysteresisMin</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>10</td>
<td>Since the percentage hysteresis can also lead to strong fluctuations when the illuminance is low, a minimum hysteresis can be specified via this property. The minimum hysteresis is an absolute value with a value range from 0 to 255. The value depends on the resolution.</td>
</tr>
<tr>
<td>nReportTimer</td>
<td>BYTE</td>
<td>Get, Set</td>
<td>30 s</td>
<td>Time (0…255 s) after which the pending event of the light sensor is repeated if no other event has occurred in the meantime.</td>
</tr>
</tbody>
</table>

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
Determine the static state from the events of a button.

For correct evaluation, the events **Button Released** (Bit 0) and **Button Pressed** (Bit 1) must be activated via the event filter. Release only those events that you need for the application. If too many events are sent, this can have a negative effect on the behavior of the application.

**VAR_INPUT**

nPushButtonEvent : UINT;

**nPushButtonEvent**: This variable contains the individual events of the push button, which are represented by the respective bits.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Push button released.</td>
</tr>
<tr>
<td>1</td>
<td>Push button pressed.</td>
</tr>
<tr>
<td>2</td>
<td>Short keystroke.</td>
</tr>
<tr>
<td>3</td>
<td>Double keystroke.</td>
</tr>
<tr>
<td>4</td>
<td>Start long keystroke.</td>
</tr>
<tr>
<td>5</td>
<td>Repeat long keystroke.</td>
</tr>
<tr>
<td>6</td>
<td>Stop long keystroke.</td>
</tr>
<tr>
<td>7</td>
<td>Push button free again; was previously blocked.</td>
</tr>
<tr>
<td>8</td>
<td>Push button blocked.</td>
</tr>
</tbody>
</table>

The function block **FB_DALI_ToPushButtonState** can be used to convert the state of a push button into a BOOL variable.

**VAR_OUTPUT**

bPushButton : BOOL;

**bPushButton**: This variable returns the static state of the button, which was determined on the basis of the individual events.

**Example**

The example shows how the events of a 4-fold DALI push button are converted into four individual variables.

**PROGRAM Demo_ST**

VAR

fbSwitchPanel : FB_DALI_Lunatone_Cross_Switch(Communication.fbKL6821Communication);
bInitialize : BOOL;
bError : BOOL;
bInitializing : BOOL;
aPushButton : ARRAY [1..4] OF BOOL;
END_VAR

**Program part in structured text**:

```plaintext
fbSwitchPanel{   bInitialize := bInitialize,
    nShortAddress := 0,
    bInitializing => bInitializing,
    bError => bError;
atoPushButtonState[1](nPushButtonEvent := fbSwitchPanel.nPushButton1Event, bPushButton => aPushButton[1]);
}
```
Program part in the Continuous Function Chart (CFC):

```
bPushButton => aPushButton[2]);
```

```
aToPushButtonState[3](nPushButtonEvent := fbSwitchPanel.nPushButton3Event,
bPushButton => aPushButton[3]);
```

```
aToPushButtonState[4](nPushButtonEvent := fbSwitchPanel.nPushButton4Event,
bPushButton => aPushButton[4]);
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.25</td>
<td>Tc3_DALI from v3.10.0.0</td>
</tr>
</tbody>
</table>

3.1.5 Events

3.1.5.1 FB_DALIGetInputNotification

The function block filters all received Input Notification Events according to certain filter criteria.

Each Input Notification contains the address of the sender and the event information (Event Info).

The address of the sender consists of two fields. The meaning of these fields is defined in the DALI control device by the event scheme (see E_DALIEventScheme[551]). The value of the event scheme can be read with the function block FB_DALI103QueryEventScheme[130] and written with the function block FB_DALI103SetEventScheme[117]. The two fields are a combination of the short address, instance number, instance type, instance group or device group.

The respective event can be determined through the event information.

The possible values of the events for the respective instance types can be found here:

Part 301 (push buttons) – introduction [288]

Part 302 (input devices, absolute encoder) - Introduction [304]

Part 303 (occupancy sensors) - introduction [313]

Part 304 (light sensors) - introduction [327]
The source of the Input Notification is defined by the inputs eEventScheme, nAddressInfo1 and nAddressInfo2. As soon as an Input Notification is received from this source, the output bNewData is set and the event information is output on the output nEventInfo.

For each Input Notification Event, which is to be received and processed further, an instance of FB_DALIGetInputNotification must be created and configured with the correct filter criteria.

Moreover, this information can also be obtained at runtime with the help of the TwinCAT Communication Monitor [572] extension.

The function block cannot be used when using the KL6811.

VAR_INPUT
bEnable : BOOL;
eEventScheme : E_DALIEventScheme := E_DALIEventScheme.DeviceInstance;
nAddressInfo1 : BYTE;
nAddressInfo2 : BYTE;

bEnable: If this input is TRUE, all received Input Notification Events of the DALI control devices are output according to the specified filter criteria. The filter criteria are defined by the inputs eEventScheme, nAddressInfo1 and nAddressInfo2.

eEventScheme: This input specifies the event scheme that the expected event must have. The event scheme also determines the meaning of the two inputs nAddressInfo1 and nAddressInfo2 (see E_DALIEventScheme [551]).

nAddressInfo1, nAddressInfo2: (see table below)

<table>
<thead>
<tr>
<th>eEventScheme</th>
<th>nAddressInfo1</th>
<th>nAddressInfo2</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_DALIEventScheme.Instance</td>
<td>Instance type (0...31)</td>
<td>Instance number (0..31)</td>
</tr>
<tr>
<td>E_DALIEventScheme.Device</td>
<td>Short address (0...63)</td>
<td>Instance type (0...31)</td>
</tr>
<tr>
<td>E_DALIEventScheme.DeviceInstance</td>
<td>Short address (0...63)</td>
<td>Instance number (0...31)</td>
</tr>
<tr>
<td>E_DALIEventScheme.DeviceGroup</td>
<td>Device group (0...31)</td>
<td>Instance type (0...31)</td>
</tr>
<tr>
<td>E_DALIEventScheme.InstanceGroup</td>
<td>Instance group (0...31)</td>
<td>Instance type (0...31)</td>
</tr>
</tbody>
</table>

VAR_OUTPUT
bError : BOOL;
ipResultMessage : I_TcMessage;
bNewData : BOOL;
nEventInfo : WORD;

bError: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.

ipResultMessage: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.

bNewData: As soon as an Input Notification Event meets the desired filter criteria, this output is set to TRUE for one PLC cycle.

nEventInfo: If the output bNewData is TRUE, further information about the event can be found at this output. The exact meaning depends on the device type and is described in the respective Part 3xx of IEC 62386.
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### 3.1.5.2 FB_DALIGetPowerCycleNotification

Filters all received Power Cycle Notification Events.

Each Power Cycle Notification Event sent by a DALI control device contains two fields that provide information about the event source. These two fields are the short address and the smallest group address to which the DALI control device belongs.

An instance of FB_DALIGetPowerCycleNotification must be created for each KL6821 through which Power Cycle Notification Events are to be received and processed. If a Power Cycle Notification Event is detected, the output bNewData is set to TRUE for one PLC cycle. The two outputs nShortAddress and nLowestDeviceGroup provide information about the exact source of the event.

The function block cannot be used when using the KL6811.

**VAR_INPUT**

- **bEnable**: If this input is TRUE, all received Power Cycle Notification Events of the DALI control devices are output.

**VAR_OUTPUT**

- **bError**: This output is switched to TRUE if an error occurs during the execution. Further information about the error can be queried via the variable ipResultMessage. The output is set to FALSE again as soon as bBusy switches to TRUE.
- **ipResultMessage**: Interface pointer (see Error evaluation [574]) that can be used to obtain detailed information about the processing of the function block (see Runtime messages [561]). The interface pointer is valid after bBusy has changed from TRUE to FALSE.
- **bNewData**: As soon as a Power Cycle Notification Event matches the desired filter criteria, this output is set to TRUE for one PLC cycle.
- **nShortAddress**: This output contains the short address (0…63) of the DALI control device that sent the Power Cycle Notification Event. If the DALI control device does not have a short address, 255 (MASK) is output.
- **nLowestDeviceGroup**: This output contains the lowest group address (0…31) of the DALI control device that sent the Power Cycle Notification Event. If the DALI control device is not assigned to a group, 255 (MASK) is output.
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3.DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### 3.2 DUTs

#### 3.2.1 Structures

##### 3.2.1.1 ST_DALIChangeAddressList

```plaintext
TYPE ST_DALIChangeAddressList:
STRUCT
  nOldAddress: BYTE;
  nNewAddress: BYTE;
  nRandomAddress: UDINT;
  nErrors: DWORD;
END_STRUCT
END_TYPE
```

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3.DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

##### 3.2.1.2 ST_DALIControlGearSettings

```plaintext
TYPE ST_DALIControlGearSettings:
STRUCT
  nErrors: DWORD;
  bPresent: BOOL;
  nActualLevel: BYTE;
  nPowerOnLevel: BYTE;
  nSystemFailureLevel: BYTE;
  nMinLevel: BYTE;
  nMaxLevel: BYTE;
  eFadeRate: E_DALIFadeRate;
  eFadeTime: E_DALIFadeTime;
  eFadeTimeBase: E_DALIExtendedFadeTimeBase;
  eFadeTimeMultiplier: E_DALIExtendedFadeTimeMultiplier;
  nRandomAddress: DWORD;
  nGroups: WORD;
  aSceneLevels: ARRAY [0..15] OF BYTE;
  nStatus: BYTE;
  nMajorVersion: BYTE;
  nMinorVersion: BYTE;
  aDeviceTypes: ARRAY [1..20] OF E_DALIDeviceType;
  nPhysicalMinLevel: BYTE;
END_STRUCT
END_TYPE
```

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3.DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

##### 3.2.1.3 ST_KL6811InData

```plaintext
TYPE ST_KL6811InData:
STRUCT
  nStatus: BYTE;
END_STRUCT
```

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3.DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
Requirements

### Development environment | required PLC library
---|---
TwinCAT from v3.1.4022.29 | Tc3_DALI from v3.1.4.0

#### 3.2.1.4 ST_KL6811OutData

```pascal
TYPE ST_KL6811OutData :
  STRUCT
    nCtrl     : BYTE;
nData     : WORD;
  END_STRUCT
END_TYPE
```

Requirements

### Development environment | required PLC library
---|---
TwinCAT from v3.1.4022.29 | Tc3_DALI from v3.1.4.0

#### 3.2.1.5 ST_KL6821InData

```pascal
TYPE ST_KL6821InData :
  STRUCT
    nStatus    : WORD;
aData      : ARRAY [0..3] OF BYTE;
  END_STRUCT
END_TYPE
```

Requirements

### Development environment | required PLC library
---|---
TwinCAT from v3.1.4022.29 | Tc3_DALI from v3.1.4.0

#### 3.2.1.6 ST_KL6821OutData

```pascal
TYPE ST_KL6821OutData :
  STRUCT
    nCtrl     : WORD;
aData      : ARRAY [0..3] OF BYTE;
  END_STRUCT
END_TYPE
```

Requirements

### Development environment | required PLC library
---|---
TwinCAT from v3.1.4022.29 | Tc3_DALI from v3.1.4.0

#### 3.2.2 Enumerations

##### 3.2.2.1 E_DALIAddressType

```pascal
TYPE E_DALIAddressType :
{
  Short    := 0,
  Group    := 1,
  Broadcast := 2,
}
```
BroadcastUnaddr := 3
) BYTE := Short;
END_TYPE

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

### 3.2.2.2 E.DALIColourTemperatureTcLimit

**TYPE E.DALIColourTemperatureTcLimit :**

| ColourTemperatureTcCoolest  | 0                 |
| ColourTemperatureTcWarmest  | 1                 |
| ColourTemperatureTcPhysicalCoolest | 2          |
| ColourTemperatureTcPhysicalWarmest | 3          |

) BYTE;
END_TYPE

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### 3.2.2.3 E.DALIColourType

**TYPE E.DALIColourType :**

| Unknown                     | 16#00 |
| ColourTemperatureTc         | 16#20 |
| Mask                        | 255   |

) BYTE := Mask;
END_TYPE

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

### 3.2.2.4 E.DALIColourValue

**TYPE E.DALIColourValue :**

| ColourTemperatureTc            | 2                 |
| ColourTemperatureTcCoolest     | 128               |
| ColourTemperatureTcPhysicalCoolest | 129        |
| ColourTemperatureTcWarmest    | 130               |
| ColourTemperatureTcPhysicalWarmest | 131        |
| TemporaryColourTemperatureTc   | 194               |
| TemporaryColourType            | 208               |
| ReportColourTemperatureTc      | 226               |
| ReportColourType               | 240               |

) BYTE;
END_TYPE

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3_DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>
3.2.2.5 E_DALICommandPriority

TYPE E_DALICommandPriority :
{
  High := 1,
  MiddleHigh := 2,
  Middle := 3,
  MiddleLow := 4,
  Low := 5,
  Unknown := 255
} BYTE := Middle;
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.2.2.6 E_DALIConfigurationCommand

TYPE E_DALIConfigurationCommand :
{
  DoNothing := 0,
  Off := 1,
  RecallMaxLevel := 2,
  RecallMinLevel := 3,
  GoToScene0 := 4,
  GoToScene1 := 5,
  GoToScene2 := 6,
  GoToScene3 := 7,
  GoToScene4 := 8,
  GoToScene5 := 9,
  GoToScene6 := 10,
  GoToScene7 := 11,
  GoToScene8 := 12,
  GoToScene9 := 13,
  GoToScene10 := 14,
  GoToScene11 := 15,
  GoToScene12 := 16,
  GoToScene13 := 17,
  GoToScene14 := 18,
  GoToScene15 := 19
};
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.2.2.7 E_DALIDataFrameType

TYPE E_DALIDataFrameType :
{
  Bit16 := 1,
  Bit24 := 3
} BYTE;
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>
3.2.2.8  E_DALIDeviceType

TYPE E_DALIDeviceType :
{
  DT00FluorescentLamp := 0,
  DT01IndependentEmergencyLighting := 1,
  DT02DischargeLamp := 2,
  DT03LowVoltageHalogenLamp := 3,
  DT04IncandescentLamp := 4,
  DT05ConversionOfDigitalSignalToDCVoltage := 5,
  DT06LEDModule := 6,
  DT07SwitchingFunction := 7,
  DT08ColorControl := 8,
  DT09Sequencer := 9,
  DT15LoadReferencing := 15,
  DT16ThermalGearProtection := 16,
  DT17DimmingCurveSelection := 17,
  DT19CentrallySuppliedEmergencyOperation := 19,
  DT20LoadShedding := 20,
  DT21ThermalLampProtection := 21,
  DT23IntegratedLightSource := 23,
  DT49IntegratedBusPowerSupply := 49,
  DT50MemoryBank1Extension := 50,
  DT51EnergyReporting := 51,
  DT52DiagnosticsMaintenance := 52,
  Undefined := 255
} BYTE := Unknown;
END_TYPE

DT00FluorescentLamp: Part 201: Standard device (device type 0)
DT01IndependentEmergencyLighting: Part 202: Device for emergency lighting (device type 1)
DT02DischargeLamp: Part 203: Device for discharge lamps (device type 2)
DT03LowVoltageHalogenLamp: Part 204: Device for low-voltage halogen lamps (device type 3)
DT04IncandescentLamp: Part 205: Supply voltage controller for incandescent lamps (device type 4)
DT05ConversionOfDigitalSignalToDCVoltage: Part 206: Device for converting digital signals to DC voltage signals (device type 5)
DT06LEDModule: Part 207: Device for LED modules (device type 6)
DT07SwitchingFunction: Part 208: Device for switching functions (device type 7)
DT08ColorControl: Part 209: Device for color/color temperature control (device type 8)
DT09Sequencer: Part 210: Sequencer (device type 9)
DT15LoadReferencing: Part 216: Load referencing (device type 15)
DT16ThermalGearProtection: Part 217: Thermal operating device protection (device type 16)
DT17DimmingCurveSelection: Part 218: Dimming curve selection (device type 17)
DT19CentrallySuppliedEmergencyOperation: Part 220: Centrally supplied emergency operation (device type 19)
DT20LoadShedding: Part 221: Power adaptation (device type 20)
DT21ThermalLampProtection: Part 222: Thermal lamp protection (device type 21)
DT23IntegratedLightSource: Part 224: Non-exchangeable light sources (device type 22)
DT49IntegratedBusPowerSupply: Part 250: Device with integrated DALI Bus power supply (device type 49)
DT50MemoryBank1Extension: Part 251: Further information and parameters for DALI control gears in memory bank 1 (device type 50)
DT51EnergyReporting: Part 252: Further parameters for the creation of an energy report (device type 51)
DT52DiagnosticsMaintenance: Part 253: Further parameters with diagnostic and maintenance information for DALI control gears (device type 52)

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.10</td>
<td>Tc3.DALI from v3.4.0.0</td>
</tr>
</tbody>
</table>

3.2.2.9    E_DALIDimmingCurve

TYPE E_DALIDimmingCurve :
{
  Standard := 0,
  Linear := 1,
  Unknown := 255
} BYTE := Unknown;
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3.DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

3.2.2.10    E_DALIEventPriority

TYPE E_DALIEventPriority :
{
  MiddleHigh := 2,
  Middle := 3,
  MiddleLow := 4,
  Low := 5,
  Unknown := 255
} BYTE := Middle;
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3.DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

3.2.2.11    E_DALIEventScheme

TYPE E_DALIEventScheme :
{
  Instance := 0,
  Device := 1,
  DeviceInstance := 2,
  DeviceGroup := 3,
  InstanceGroup := 4,
  Unknown := 255
} BYTE := Unknown;
END_TYPE

| Instance: Instance addressing with instance type and instance number. |
| Device: Device addressing with short address and instance type. |
| DeviceInstance: Device/instance addressing with short address and instance number. |
| DeviceGroup: Addressing of device groups with device group and instance type. |
| InstanceGroup: Addressing of instance groups with instance group and instance type. |
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.2.2.12 E_DALIEventType

```
TYPE E_DALIEventType :
{
    InputNotification := 0,
    PowerNotification := 1,
    Unknown := 255
} BYTE := Unknown;
END_TYPE
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

3.2.2.13 E_DALIExtendedFadeTimeBase

```
TYPE E_DALIExtendedFadeTimeBase :
{
    Base01 := 0,
    Base02 := 1,
    Base03 := 2,
    Base04 := 3,
    Base05 := 4,
    Base06 := 5,
    Base07 := 6,
    Base08 := 7,
    Base09 := 8,
    Base10 := 9,
    Base11 := 10,
    Base12 := 11,
    Base13 := 12,
    Base14 := 13,
    Base15 := 14,
    Base16 := 15,
    Unknown := 255
} BYTE := Unknown;
END_TYPE
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

3.2.2.14 E_DALIExtendedFadeTimeMultiplier

```
TYPE E_DALIExtendedFadeTimeMultiplier :
{
    Disabled := 0,
    Multiplier100ms := 1,
    Multipler1s := 2,
    Multiplier10s := 3,
    Multiplier1min := 4,
    Unknown := 255
} BYTE := Unknown;
END_TYPE
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>
3.2.2.15  E_DALIFadeRate

TYPE E_DALIFadeRate :
{
  N358StepsPerSec := 1,
  N253StepsPerSec := 2,
  N179StepsPerSec := 3,
  N127StepsPerSec := 4,
  N089StepsPerSec := 5,
  N063StepsPerSec := 6,
  N045StepsPerSec := 7,
  N032StepsPerSec := 8,
  N022StepsPerSec := 9,
  N016StepsPerSec := 10,
  N011StepsPerSec := 11,
  N008StepsPerSec := 12,
  N006StepsPerSec := 13,
  N004StepsPerSec := 14,
  N003StepsPerSec := 15,
  Unknown := 255
} BYTE := Unknown;
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

3.2.2.16  E_DALIFadeTime

TYPE E_DALIFadeTime :
{
  Disabled := 0,
  T00707ms := 1,
  T01000ms := 2,
  T01400ms := 3,
  T02000ms := 4,
  T02800ms := 5,
  T04000ms := 6,
  T05700ms := 7,
  T08000ms := 8,
  T11300ms := 9,
  T16000ms := 10,
  T22600ms := 11,
  T32000ms := 12,
  T45300ms := 13,
  T64000ms := 14,
  T90500ms := 15,
  Unknown := 255
} BYTE := Unknown;
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.4</td>
<td>Tc3_DALI from v3.2.0.0</td>
</tr>
</tbody>
</table>

3.2.2.17  E_DALIFastFadeTime

TYPE E_DALIFastFadeTime :
{
  Disabled := 0,
  T100ms := 4,
  T200ms := 8,
  T225ms := 9,
  T300ms := 12,
  T400ms := 16,
  T500ms := 20,
  T600ms := 24,
  T700ms := 27,
}
Unknown := 255
) BYTE := Unknown;
END_TYPE

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4024.11</td>
<td>Tc3_DALI from v3.5.0.0</td>
</tr>
</tbody>
</table>

#### 3.2.2.18 E_DALIInstanceAddressType

TYPE E_DALIInstanceAddressType :
|
| InstanceNumber := 0, |
| InstanceGroup := 1, |
| InstanceType := 2, |
| FeatureNumber := 3, |
| FeatureGroup := 4, |
| FeatureType := 5, |
| FeatureBroadcast := 6, |
| InstanceBroadcast := 7, |
| FeatureDevice := 8, |
| Device := 9 |
) BYTE := InstanceNumber;
END_TYPE

**InstanceNumber**: Instance number (0...31).

**InstanceGroup**: Instance group (0...31).

**InstanceType**: Instance type (0...31).

**FeatureNumber**: Feature at instance number level (0...31).

**FeatureGroup**: Feature at instance group level (0...31).

**FeatureType**: Feature at instance type level (0...31).

**FeatureBroadcast**: Feature at instance broadcast level.

**InstanceBroadcast**: Instance broadcast.

**FeatureDevice**: Feature at device level.

**Device**: Device

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT from v3.1.4022.29</td>
<td>Tc3_DALI from v3.1.4.0</td>
</tr>
</tbody>
</table>

#### 3.2.2.19 E_DALIInstanceType

TYPE E_DALIInstanceType :
|
| IT00GenericInstance := 0, |
| IT01PushButton := 1, |
| IT02AbsoluteInputDevice := 2, |
| IT03OccupancySensor := 3, |
| IT04LightSensor := 4, |
| IT05ColourSensor := 5, |
| IT06GeneralPurposeSensor := 6, |
| IT32InputControlDeviceFeedback := 32, |
| IT33ManualConfiguration := 33, |
| IT51LuminaireMountedControlDevice := 51, |
| Unknown := 255 |
) BYTE := Unknown;
END_TYPE

**IT00GenericInstance**: Part 103: Control device (instance type 0)
ITE1000
Version: 1.10

IT01PushButton: Part 301: Push button (instance type 1)

IT02AbsoluteInputDevice: Part 302: Absolute input device (instance type 2)

IT03OccupancySensor: Part 303: Occupancy sensor (instance type 3)

IT04LightSensor: Part 304: Light sensor (instance type 4)

IT05ColourSensor: Part 305: Color sensor (instance type 5)

IT06GeneralPurposeSensor: Part 306: Universal sensor (instance type 6)

IT32InputControlDeviceFeedback: Part 332: Feedback for input devices (instance type 32)

IT33ManualConfiguration: Part 333: Manual configuration (instance type 33)

IT51LuminaireMountedControlDevice: Part 351: Control device integrated in the luminaire (instance type 51)

Requirements

Development environment | Required PLC library
------------------------|--------------------------
TwinCAT from v3.1.4024.11 | Tc3_DALI from v3.5.0.0

3.2.2.20 E_DALI_LoytecInputConfiguration

TYPE E_DALI_LoytecInputConfiguration :
{
  PushButton := 0,
  GenericInput := 1,
  AbsoluteValue := 2
} BYTE := PushButton;
END_TYPE

Requirements

Development environment | Required PLC library
------------------------|--------------------------
TwinCAT from v3.1.4024.10 | Tc3_DALI from v3.4.0.0

3.2.2.21 E_DALI_LoytecResistanceConfiguration

TYPE E_DALI_LoytecResistanceConfiguration :
{
  Resistance1kOhm := 0,
  Resistance10kOhm := 1,
} BYTE := Resistance10kOhm;
END_TYPE

Requirements

Development environment | Required PLC library
------------------------|--------------------------
TwinCAT from v3.1.4024.10 | Tc3_DALI from v3.4.0.0

3.3 GVLs

3.3.1 Constants

VAR_GLOBAL
eEventTraceLevel                   : TcEventSeverity := TcEventSeverity.Critical;
END_VAR

VAR_GLOBAL_CONSTANT
cMemoryMode                       : DWORD := 16#0000_0001;
The library contains some parameters which can be used to change general library settings. The parameter list can be accessed via the Library Manager. The values of the individual parameters can also be adjusted there.

### NOTE

Note that some parameters have a profound effect on the way the library works.
3.4 Integration into TwinCAT

3.4.1 KL6821 with CX5120

This sample explains how to write a simple PLC program for DALI in TwinCAT and how to link it with the hardware.

A single lamp is to be controlled and switched to the maximum output value or switched off with a push button.

Sample: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_DALI/Resources/zip/27021603775323659.zip

Hardware

Setting up the components

- 1x Embedded PC CX5120
- 1x KL1104 digital 4-channel input terminal (for the switch-on/switch-off function)
- 1x DALI terminal KL6821
- 1x KL9010 end terminal

Set up the hardware and the DALI components as described in the documentation.

This sample assumes that an On button was connected to the first KL1104 input and an Off button to the second. There is a lamp at DALI address 0.

Software

Creation of the PLC program

Create a new TwinCAT XAE project and a Standard PLC project. Add the library Tc3_DALI in the PLC project under References. Generate a global variable list with the name GVL_DALI and create the following variables:

```
VAR_GLOBAL
  bSwitchOn AT %I*: BOOL;
  bSwitchOff AT %I*: BOOL;
  stKL6821InData AT %I*: ST_KL6821InData;
  stKL6821OutData AT %Q*: ST_KL6821OutData;
END_VAR
```

bSwitchOn: Input variable for the On button.

bSwitchOff: Input variable for the Off button.

stKL6821InData: Input variable for the DALI terminal (see ST_KL6821InData [547]).

stKL6821OutData: Output variable for the DALI terminal (see ST_KL6821OutData [547]).

Create a program (CFC) for the background communication with DALI. The function block FB_KL6821Communication [343] is called in the program. In the communication block ensure that the structures stInData and stOutData are linked.
Create a MAIN program (CFC) in which the function blocks FB_DALI102RecallMaxLevel \([\text{62}]\) and FB_DALI102Off \([\text{60}]\) are declared as follows.

```plaintext
PROGRAM MAIN
VAR
  fb102RecallMaxLevel : FB_DALI102RecallMaxLevel(Communication.fbKL6821Communication);
  fb102Off : FB_DALI102Off(Communication.fbKL6821Communication);
END_VAR
```

The communication block is specified in the round brackets after the declaration. The reference to the desired DALI terminal is defined via this specification.

Call the two instances of the function blocks FB_DALI102RecallMaxLevel and FB_DALI102Off with the following variables.

The input \(bStart\) of the function block for switching on a lamp with the maximum output value is linked to the global variable \(bSwitchOn\).

The input \(bStart\) of the function block for switching off a lamp is linked to the global variable \(bSwitchOff\).

Navigate to the task configuration section and configure the PlcTask. By way of example, the task is assigned priority 16 and a cycle time of 6 ms.
Create a further task for the background communication. Assign a higher priority (smaller number) and a lower interval time to this task than the PlcTask.

Add the program for the communication to this task. Further information on task configuration can be found in the description of the function block FB_KL6821Communication [p. 343].

**I/O configuration**

Select the CX as target system and initiate a search for its hardware. In the project instance within the PLC section, you can see that the input and output variables are assigned to the corresponding tasks.
Now link the global variables of PLC program with the inputs and outputs of the Bus Terminals. Create the Solution and enable the configuration.

The lamp with the maximum brightness value is switched on by pressing the first push button. The second push button can be used to switch it off again.
3.5 Runtime messages
<table>
<thead>
<tr>
<th>Value (hex)</th>
<th>Value (dec)</th>
<th>Text ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0</td>
<td>NoError</td>
<td>No error.</td>
</tr>
<tr>
<td>0x0001</td>
<td>1</td>
<td>NoResponseFromTheDALITerminal</td>
<td>No response from the DALI terminal.</td>
</tr>
<tr>
<td>0x0002</td>
<td>2</td>
<td>NoResponseFromTheDALIDevice</td>
<td>No response from the DALI device.</td>
</tr>
<tr>
<td>0x0003</td>
<td>3</td>
<td>CommandBufferOverflow</td>
<td>Overflow of the command buffer.</td>
</tr>
<tr>
<td>0x0004</td>
<td>4</td>
<td>NoAnswerFromTheCommunicationBlock</td>
<td>No response from the communication block.</td>
</tr>
<tr>
<td>0x0005</td>
<td>5</td>
<td>DALICollisionDetectedOnTheBackwardChannel</td>
<td>DALI collision detected on the backward channel: During the transfer of a DALI telegram, a collision with the send data of another DALI slave was detected.</td>
</tr>
<tr>
<td>0x0006</td>
<td>6</td>
<td>DALICollisionDetectedOnTheForwardChannel</td>
<td>DALI collision detected on the forward channel: During the transfer of a DALI telegram, a collision with the send data of another DALI master was detected. The error also occurs as soon as the 24 V supply is missing at the power contacts of the KL6811.</td>
</tr>
<tr>
<td>0x0007</td>
<td>7</td>
<td>OverloadOfTheInternalDALIPowerSupplyUnitOfTheBusTerminal_BusUnderVoltage</td>
<td>When using the internal DALI power supply of the Bus Terminal: overload of the internal DALI power supply of the Bus Terminal (bus undervoltage).</td>
</tr>
<tr>
<td>0x0008</td>
<td>8</td>
<td>PowerSupplyUnitFaultDetected</td>
<td>When using the internal DALI power supply unit: Power supply unit fault detected.</td>
</tr>
<tr>
<td>0x0009</td>
<td>9</td>
<td>TheProcessImageWasDisabledByDI1OrDI2</td>
<td>The process image of the KL6821 was disabled by the DI1 or DI2 inputs of the terminal.</td>
</tr>
<tr>
<td>0x000A</td>
<td>10</td>
<td>ShortCircuitDetectedOnTheDALIBus</td>
<td>The 24 V DC supply voltage at connections 1 and 5 of the KL6821 is missing, or a short circuit has been detected on the DALI bus.</td>
</tr>
<tr>
<td>0x000B</td>
<td>11</td>
<td>CollisionErrorOnTheDALIBus</td>
<td>A collision on the DALI bus was detected while a DALI command was sent.</td>
</tr>
<tr>
<td>0x000C</td>
<td>12</td>
<td>ItemInReceiveBufferIsInvalid</td>
<td>The entry in the receive buffer of the KL6821 is invalid.</td>
</tr>
<tr>
<td>0x000D</td>
<td>13</td>
<td>TheInterfaceToTheCommunicationBufferIsNotInitialized</td>
<td>The interface pointer to the communication buffer is not initialized.</td>
</tr>
<tr>
<td>0x000E</td>
<td>14</td>
<td>TheCommandBuffersHaveBeenBlockedForLongerThanPermitted</td>
<td>The command buffer has been blocked for longer than permitted.</td>
</tr>
<tr>
<td>0x000F</td>
<td>15</td>
<td>TheTerminalHasReturnedAnErrorDuringInternalAddressing</td>
<td>During internal addressing, the terminal has returned an error.</td>
</tr>
<tr>
<td>0x0010</td>
<td>16</td>
<td>DuringInternalAddressingTheTerminalHasDetectedAShortCircuitOnTheBus</td>
<td>During internal addressing, the terminal has detected a short circuit on the bus.</td>
</tr>
<tr>
<td>0x0011</td>
<td>17</td>
<td>DuringInternalAddressingTheTerminalHasDetectedThatThereIsNoFurtherShortAddressAvailable</td>
<td>During internal addressing, the terminal has detected that there is no free short address available.</td>
</tr>
<tr>
<td>0x0012</td>
<td>18</td>
<td>DuringInternalAddressingTheTerminalHasDetectedThatSeveralDevicesHaveTheSameLongAddress</td>
<td>During internal addressing, the terminal has detected that several devices have the same long address.</td>
</tr>
<tr>
<td>0x0013</td>
<td>19</td>
<td>InternalAddressingHasFailed3Times</td>
<td>The internal addressing failed three times.</td>
</tr>
<tr>
<td>0x0014</td>
<td>20</td>
<td>TimeoutDuringInternalAddressing</td>
<td>Timeout for internal addressing. The terminal has not sent a response following the start of internal addressing.</td>
</tr>
<tr>
<td>0x0015</td>
<td>21</td>
<td>NoResponseFromTheDALIDeviceInstance</td>
<td>No response from the instance of the DALI device.</td>
</tr>
<tr>
<td>0x0016</td>
<td>22</td>
<td>TheKL6811DoesNotSupportThisDALICommand</td>
<td>The KL6811 supports only 16-bit commands (IEC 62386-102 and IEC 62386-2xx).</td>
</tr>
<tr>
<td>0x0017</td>
<td>23</td>
<td>ErrorDuringTheConfigurationOfTheTerminal</td>
<td>An error occurred during configuration of the terminal.</td>
</tr>
<tr>
<td>0x0018</td>
<td>24</td>
<td>NoResponseFromTheDALIDevice2</td>
<td>No response from the DALI device.</td>
</tr>
<tr>
<td>0x0019</td>
<td>25</td>
<td>NoFreeShortAddressAvailable</td>
<td>There is no further short address available.</td>
</tr>
<tr>
<td>0x001A</td>
<td>26</td>
<td>ErrorDuringAddressing</td>
<td>An error occurred during addressing.</td>
</tr>
<tr>
<td>0x00B9</td>
<td>185</td>
<td>ParameterSensitivityOccupancyPartial3Part303LiesOutsideOfTheValidRange</td>
<td>Parameter nSensitivityN lies outside the valid range (1…100).</td>
</tr>
<tr>
<td>0x00BA</td>
<td>186</td>
<td>ParameterSensitivityOccupancyPartial2Part303LiesOutsideOfTheValidRange</td>
<td>Parameter nSensitivityN lies outside the valid range (0…3).</td>
</tr>
<tr>
<td>0x00BB</td>
<td>187</td>
<td>ParameterExtendedFadeTimeBaseLiesOutsideOfTheValidRange</td>
<td>Parameter eExtendedFadeTimeBase lies outside the valid range (0…15).</td>
</tr>
<tr>
<td>0x00BC</td>
<td>188</td>
<td>ParameterExtendedFadeTimeMultiplierLiesOutsideOfTheValidRange</td>
<td>Parameter eExtendedFadeTimeMultiplier lies outside the valid range (0…4).</td>
</tr>
<tr>
<td>0x00BD</td>
<td>189</td>
<td>ParameterDeadtimeLiesOutsideOfTheValidRange</td>
<td>Parameter Ddeadtime lies outside of the valid range (T#3S…T#600S).</td>
</tr>
<tr>
<td>0x00CB</td>
<td>190</td>
<td>ParameterHysteresisLiesOutsideOfTheValidRange2</td>
<td>Parameter nHysteresis lies outside of the valid range (0…25 %).</td>
</tr>
<tr>
<td>Value (hex)</td>
<td>Value (dec)</td>
<td>Text ID</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x00CC</td>
<td>191</td>
<td>ParameterDeadtimeTimerN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DeadtimeTimer lies outside the valid range (0...12750 ms).</td>
</tr>
<tr>
<td>0x00CD</td>
<td>192</td>
<td>ParameterHoldTimerN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$HoldTimer lies outside of the valid range (1...2538 ms).</td>
</tr>
<tr>
<td>0x00C1</td>
<td>193</td>
<td>ParameterStuckTimerN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$StuckTimer lies outside the valid range (5...255 s).</td>
</tr>
<tr>
<td>0x00C2</td>
<td>194</td>
<td>ParameterShortN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$Short lies outside the valid range ($t$ShortMin...255).</td>
</tr>
<tr>
<td>0x00C3</td>
<td>195</td>
<td>ParameterShortTimerPart301N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$ShortTimerN lies outside the valid range ($t$ShortMin...5100 ms).</td>
</tr>
<tr>
<td>0x00C4</td>
<td>196</td>
<td>ParameterShortTimerN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$ShortTimer lies outside the valid range ($t$ShortMin...5100 ms).</td>
</tr>
<tr>
<td>0x00C5</td>
<td>197</td>
<td>ParameterDoubleTimerPart301N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DoubleTimerN lies outside of the valid range (0, $t$DoubleMin...2000 ms).</td>
</tr>
<tr>
<td>0x00C6</td>
<td>198</td>
<td>ParameterDoubleTimerN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DoubleTimer lies outside of the valid range (0, $t$DoubleMin...2000 ms).</td>
</tr>
<tr>
<td>0x00C7</td>
<td>199</td>
<td>ParameterRepeatTimerN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$RepeatTimer lies outside the valid range (100...2000 ms).</td>
</tr>
<tr>
<td>0x00C8</td>
<td>200</td>
<td>ParameterEventSchemeN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$EventScheme lies outside of the valid range (0...4).</td>
</tr>
<tr>
<td>0x00C9</td>
<td>201</td>
<td>ParameterAddressIsAGroupAddressAndLiesOutsideOfTheValidRange2</td>
<td>Parameter $n$Address is a group address and lies outside the valid range (0...31).</td>
</tr>
<tr>
<td>0x00CA</td>
<td>202</td>
<td>ParameterEventPriorityDeviceN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$EventPriorityN lies outside of the valid range (E_DALIEventPriority.Low...E_DALIEventPriority.MiddleHigh).</td>
</tr>
<tr>
<td>0x00CB</td>
<td>203</td>
<td>ParameterALSSxCorrectionFactorPart304N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$ALSSxCorrectionFactorN lies outside the valid range (1...50).</td>
</tr>
<tr>
<td>0x00CC</td>
<td>204</td>
<td>ParameterRoomCorrectionFactorPart304N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$RoomCorrectionFactorN lies outside the valid range (5...200).</td>
</tr>
<tr>
<td>0x00CD</td>
<td>205</td>
<td>ParameterSwitchOnLevelN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$SwitchOnLevel lies outside the valid range ($n$MinLevel...$n$MaxLevel).</td>
</tr>
<tr>
<td>0x00CE</td>
<td>206</td>
<td>ParameterHysteresisPart304N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$HysteresisN lies outside of the valid range (0...25 %).</td>
</tr>
<tr>
<td>0x00CF</td>
<td>207</td>
<td>ParameterDeadtimeTimerPart304N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DeadtimeTimerN lies outside the valid range (0...12750 ms).</td>
</tr>
<tr>
<td>0x00D0</td>
<td>208</td>
<td>ParameterSensitivityOccupancyPercentPart303N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$SensitivityN lies outside the valid range (1...5).</td>
</tr>
<tr>
<td>0x00D1</td>
<td>209</td>
<td>ParameterSensitivityOccupancyPercentPart303N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$SensitivityN lies outside the valid range (0...100 %).</td>
</tr>
<tr>
<td>0x00D2</td>
<td>210</td>
<td>ParameterDetectionRangeOccupancyPart303N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DetectionRangeN lies outside the valid range (0...100 %).</td>
</tr>
<tr>
<td>0x00D3</td>
<td>211</td>
<td>ParameterHoldTimerPart303N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$HoldTimerN lies outside of the valid range (1...2538 ms).</td>
</tr>
<tr>
<td>0x00D4</td>
<td>212</td>
<td>ParameterDeadtimeTimerPart303N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DeadtimeTimerN lies outside the valid range (0...12750 ms).</td>
</tr>
<tr>
<td>0x00D5</td>
<td>213</td>
<td>ParameterHysteresisOutsideOfTheValidRange3</td>
<td>Parameter $n$Hysteresis lies outside of the valid range (1...65535).</td>
</tr>
<tr>
<td>0x00D6</td>
<td>214</td>
<td>ParameterDeadtimeTimerPart302N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$DeadtimeTimerN lies outside the valid range (0...12750 ms).</td>
</tr>
<tr>
<td>0x00D7</td>
<td>215</td>
<td>ParameterStuckTimerPart301N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$StuckTimerN lies outside the valid range (5...255 ms).</td>
</tr>
<tr>
<td>0x00D8</td>
<td>216</td>
<td>ParameterRepeatTimerPart301N LiesOutsideOfTheValidRange</td>
<td>Parameter $n$RepeatTimerN lies outside the valid range (100...2000 ms).</td>
</tr>
<tr>
<td>0x00D9</td>
<td>217</td>
<td>ParameterUpSwitchOFTresholdN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$UpSwitchOFTresholdN lies outside the valid range (1...255).</td>
</tr>
<tr>
<td>0x00DA</td>
<td>218</td>
<td>ParameterUpSwitchOnThresholdN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$UpSwitchOnThresholdN lies outside the valid range (1...255).</td>
</tr>
<tr>
<td>0x00DB</td>
<td>219</td>
<td>ParameterLevelMemoryModeN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$LevelMemoryModeN lies outside the valid range (1...254).</td>
</tr>
<tr>
<td>0x00DC</td>
<td>220</td>
<td>ParameterMinLevelN Is Greater Than $MaxLevel$</td>
<td>Parameter $n$Min level is greater than $n$MaxLevel.</td>
</tr>
<tr>
<td>0x00DD</td>
<td>221</td>
<td>ParameterReferenceDeviceAddressN LiesOutsideOfTheValidRange</td>
<td>Parameter $n$ReferenceDeviceAddressN lies outside the valid range (0...63).</td>
</tr>
<tr>
<td>0x00DE</td>
<td>222</td>
<td>ErrorDuringReadingOffsetFromMemoryBank</td>
<td>An error occurred while reading an offset from the memory bank.</td>
</tr>
<tr>
<td>Value (hex)</td>
<td>Value (dec)</td>
<td>Text ID</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x00DF</td>
<td>223</td>
<td>ParameterSubRangeStartLiesOutsideOfTheValidRange</td>
<td>Parameter nSubRangeStart lies outside the valid range.</td>
</tr>
<tr>
<td>0x00E0</td>
<td>224</td>
<td>ErrorReadingMemoryBankDTR0IsWrong</td>
<td>An error occurred while reading out a memory bank. The value of DTR0 is not as expected.</td>
</tr>
<tr>
<td>0x00E1</td>
<td>225</td>
<td>ParameterSubRangeEndLiesOutsideOfTheValidRange</td>
<td>Parameter nSubRangeEnd lies outside the valid range.</td>
</tr>
<tr>
<td>0x00E2</td>
<td>226</td>
<td>ReadCommandReturnedTMASK</td>
<td>The DALI device returns TMASK. The value is not currently available.</td>
</tr>
<tr>
<td>0x00E3</td>
<td>227</td>
<td>ReadCommandReturnedMASK</td>
<td>The DALI device returns MASK. The value is not available.</td>
</tr>
<tr>
<td>0x00E4</td>
<td>228</td>
<td>ParameterInternalControlGearReferenceTemperatureLiesOutsideOfTheValidRange</td>
<td>Parameter nInternalControlGearReferenceTemperature lies outside the valid range (-60…193 °C).</td>
</tr>
<tr>
<td>0x00E5</td>
<td>229</td>
<td>ParameterLightSourceOnTimeResettableLiesOutsideOfTheValidRange</td>
<td>Parameter nLightSourceOnTimeResettable lies outside the valid range (0…4294967293 s).</td>
</tr>
<tr>
<td>0x00E6</td>
<td>230</td>
<td>ParameterLightSourceStartCounterResettableLiesOutsideOfTheValidRange</td>
<td>Parameter nLightSourceStartCounterResettable lies outside the valid range (0…16777213).</td>
</tr>
<tr>
<td>0x00E7</td>
<td>231</td>
<td>ParameterRatedMedianUsefulLifeOfLuminaireLiesOutsideOfTheValidRange</td>
<td>Parameter nRatedMedianUsefulLifeOfLuminaire lies outside the valid range (0…253).</td>
</tr>
<tr>
<td>0x00E8</td>
<td>232</td>
<td>ParameterRatedMedianUsefulLightSourceStartsLiesOutsideOfTheValidRange</td>
<td>Parameter nRatedMedianUsefulLightSourceStarts lies outside the valid range (0…65533)</td>
</tr>
<tr>
<td>0x00E9</td>
<td>233</td>
<td>ParameterInstanceNumberLiesOutsideOfTheValidRange</td>
<td>Parameter nInstanceNumber lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x00EA</td>
<td>234</td>
<td>ParameterShortAddressLiesOutsideOfTheValidRange2</td>
<td>Parameter nShortAddress lies outside the valid range (0…63).</td>
</tr>
<tr>
<td>0x00EB</td>
<td>235</td>
<td>ParameterSubRangeEndIsLessThanSubRangeStart</td>
<td>Parameter nSubRangeEnd is smaller than nSubRangeStart.</td>
</tr>
<tr>
<td>0x00EE</td>
<td>238</td>
<td>ParameterEventPriorityLiesOutsideOfTheValidRange</td>
<td>Parameter nEventPriority lies outside of the valid range (E_DALIEventPriority.MiddleHigh…E_DALIEventPriority.Low).</td>
</tr>
<tr>
<td>0x00EF</td>
<td>239</td>
<td>ParameterShortAddressLiesOutsideOfTheValidRange</td>
<td>Parameter nShortAddress lies outside the valid range (0…63, 255).</td>
</tr>
<tr>
<td>0x00F0</td>
<td>240</td>
<td>ParameterAddressInfoInstanceGroupLiesOutsideOfTheValidRange</td>
<td>Parameter nAddressInfo1/nAddressInfo2 defines an instance group and lies outside of the valid range (0…31).</td>
</tr>
<tr>
<td>0x00F1</td>
<td>241</td>
<td>ParameterAddressInfoDeviceGroupLiesOutsideOfTheValidRange</td>
<td>Parameter nAddressInfo1/nAddressInfo2 defines a device group and lies outside of the valid range (0…31).</td>
</tr>
<tr>
<td>0x00F2</td>
<td>242</td>
<td>ParameterAddressInfoShortAddressLiesOutsideOfTheValidRange</td>
<td>Parameter nAddressInfo1/nAddressInfo2 defines a short address and lies outside of the valid range (0…63).</td>
</tr>
<tr>
<td>0x00F3</td>
<td>243</td>
<td>ParameterAddressInfoInstanceNumberLiesOutsideOfTheValidRange</td>
<td>Parameter nAddressInfo1/nAddressInfo2 defines an instance number and lies outside of the valid range (0…31).</td>
</tr>
<tr>
<td>0x00F4</td>
<td>244</td>
<td>ParameterAddressInfoInstanceTypeLiesOutsideOfTheValidRange</td>
<td>Parameter nAddressInfo1/nAddressInfo2 defines an instance type and lies outside of the valid range (0…31).</td>
</tr>
<tr>
<td>0x00F5</td>
<td>245</td>
<td>ParameterHysteresisLiesOutsideOfTheValidRange</td>
<td>Parameter nHysteresis lies outside the valid range (0…25).</td>
</tr>
<tr>
<td>0x00F6</td>
<td>246</td>
<td>ParameterHoldLiesOutsideOfTheValidRange</td>
<td>Parameter nHold lies outside the valid range (0…254).</td>
</tr>
<tr>
<td>0x00F7</td>
<td>247</td>
<td>ParameterDoubleLiesOutsideOfTheValidRange</td>
<td>Parameter nDouble lies outside the valid range (0, tDoubleMin…100).</td>
</tr>
<tr>
<td>0x00F8</td>
<td>248</td>
<td>ParameterRepeatLiesOutsideOfTheValidRange</td>
<td>Parameter nRepeat lies outside the valid range (5…100).</td>
</tr>
<tr>
<td>0x00F9</td>
<td>249</td>
<td>ParameterStuckLiesOutsideOfTheValidRange</td>
<td>Parameter nStuck lies outside the valid range (5…255).</td>
</tr>
<tr>
<td>0x00FA</td>
<td>250</td>
<td>ParameterResolutionLiesOutsideOfTheValidRange</td>
<td>Parameter nResolution lies outside the valid range (1…64).</td>
</tr>
<tr>
<td>0x00FB</td>
<td>251</td>
<td>ParameterEventFilterLiesOutsideOfTheValidRange</td>
<td>Parameter nEventFilter lies outside of the valid range (16#00_0000…16#FF_FFFF).</td>
</tr>
<tr>
<td>0x00FC</td>
<td>252</td>
<td>ParameterChangeAddressListIsEmpty</td>
<td>Parameter aChangeAddressList is empty.</td>
</tr>
<tr>
<td>0x00FD</td>
<td>253</td>
<td>ParameterChangeAddressListContainsAnInvalidShortAddressEntry</td>
<td>Parameter aChangeAddressList contains an invalid short address entry.</td>
</tr>
<tr>
<td>0x00FE</td>
<td>254</td>
<td>ParameterChangeAddressListContainsADoubleListEntryInTheShortAddresses</td>
<td>Parameter aChangeAddressList contains a duplicate list item for the short addresses.</td>
</tr>
<tr>
<td>Value (hex)</td>
<td>Value (dec)</td>
<td>Text ID</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x00FF</td>
<td>255</td>
<td>0x0100</td>
<td>Parameter <code>ChangeAddressListContainsAnEntryForANewShortAddressWhichIsAlreadyAssigned</code> lies outside the valid range (2…250).</td>
</tr>
<tr>
<td>0x0100</td>
<td>256</td>
<td>0x0101</td>
<td>Parameter <code>MaxResponseTableEntries</code> lies outside the valid range (2…250).</td>
</tr>
<tr>
<td>0x0101</td>
<td>257</td>
<td>0x0102</td>
<td>Parameter <code>MaxEventTableEntries</code> lies outside the valid range (2…250).</td>
</tr>
<tr>
<td>0x0102</td>
<td>258</td>
<td>0x0103</td>
<td>Parameter <code>MaxCommandBufferEntries</code> lies outside the valid range (2…250).</td>
</tr>
<tr>
<td>0x0103</td>
<td>259</td>
<td>0x0104</td>
<td>Parameter <code>DataFrameType</code> lies outside the valid range.</td>
</tr>
<tr>
<td>0x0104</td>
<td>260</td>
<td>0x0105</td>
<td>Parameter <code>AddressType</code> lies outside the valid range.</td>
</tr>
<tr>
<td>0x0105</td>
<td>261</td>
<td>0x0106</td>
<td>Parameter <code>AddressIsAShortAddressAndLiesOutsideTheValidRange</code> lies outside the valid range (0…63).</td>
</tr>
<tr>
<td>0x0106</td>
<td>262</td>
<td>0x0107</td>
<td>Parameter <code>AddressIsAGroupAddressAndLiesOutsideTheValidRange</code> lies outside the valid range (0…15).</td>
</tr>
<tr>
<td>0x0107</td>
<td>263</td>
<td>0x0108</td>
<td>Parameter <code>CommandPriority</code> lies outside the valid range.</td>
</tr>
<tr>
<td>0x0108</td>
<td>264</td>
<td>0x0109</td>
<td>Parameter <code>InstanceAddressTypeLiesOutsideTheValidRange</code> lies outside the valid range.</td>
</tr>
<tr>
<td>0x0109</td>
<td>265</td>
<td>0x010A</td>
<td>Parameter <code>InstanceAddressIsAnInstanceNumberAndLiesOutsideTheValidRange</code> lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x010A</td>
<td>266</td>
<td>0x010B</td>
<td>Parameter <code>InstanceAddressIsAnInstanceGroupAndLiesOutsideTheValidRange</code> lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x010B</td>
<td>267</td>
<td>0x010C</td>
<td>Parameter <code>InstanceAddressIsAnInstanceTypeAndLiesOutsideTheValidRange</code> lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x010C</td>
<td>268</td>
<td>0x010D</td>
<td>Parameter <code>InstanceAddressIsAFeatureOnInstanceNumberLevelAndLiesOutsideTheValidRange</code> lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x010D</td>
<td>269</td>
<td>0x010E</td>
<td>Parameter <code>InstanceAddressIsAFeatureOnInstanceGroupLevelAndLiesOutsideTheValidRange</code> lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x010E</td>
<td>270</td>
<td>0x010F</td>
<td>Parameter <code>InstanceAddressIsAFeatureOnInstanceTypeLevelAndLiesOutsideTheValidRange</code> lies outside the valid range (0…31).</td>
</tr>
<tr>
<td>0x010F</td>
<td>271</td>
<td>0x0110</td>
<td>Parameter <code>Group</code> lies outside the valid range (0…15).</td>
</tr>
<tr>
<td>0x0110</td>
<td>272</td>
<td>0x0111</td>
<td>Parameter <code>Scene</code> lies outside the valid range (0…15).</td>
</tr>
<tr>
<td>0x0111</td>
<td>273</td>
<td>0x0112</td>
<td>Parameter <code>FadeTime</code> lies outside the valid range (0…15).</td>
</tr>
<tr>
<td>0x0112</td>
<td>274</td>
<td>0x0113</td>
<td>Parameter <code>FadeRate</code> lies outside the valid range (1…15).</td>
</tr>
<tr>
<td>0x0113</td>
<td>275</td>
<td>0x0114</td>
<td>Parameter <code>NewShortAddress</code> lies outside the valid range (0…63, 255).</td>
</tr>
<tr>
<td>0x0114</td>
<td>276</td>
<td>0x0115</td>
<td>Parameter <code>StartWithShortAddress</code> lies outside the valid range (0…63).</td>
</tr>
</tbody>
</table>
4 Appendix

4.1 Commissioning and diagnosis

4.1.1 KS2000

The KS2000 provides dialogs for commissioning and diagnosis of DALI devices on the KL6821.

The dialogs offer not only functions for addressing control gears and control devices, but also for writing and reading parameters. The search function automatically detects the device types and displays them in a tree structure.

All parameters of the KL6821 can also be set via the KS2000. This makes it possible, for example, to define the DALI commands that are sent when the digital inputs on the KL6821 are actuated.
Overview of the most important functions in the KS2000 for the KL6821:

- Find DALI devices
- Addressing the DALI devices, including adapting the short addresses
- Scene and group assignment of control gears
- Configuration of the control gears, including the parameters for the different device types
- Configuration of the control devices, including the parameters for the different sensor types
- Writing/reading of the memory banks
- Definition of DALI commands for DI1 and DI2 (separate for rising and falling edge)
- Definition of the DALI command for the K-bus watchdog (K-bus failure)
- Switching the internal DALI power supply unit on/off
- Activation/deactivation of the blocking of the process image for the PLC as soon as a DALI command is sent by actuating a digital input on the KL6821.

4.1.2 PLC HMI

The PLC project **Tc3_DALI_Commissioning** offers the user the possibility to configure DALI devices and the KL6821 with the aid of the PLC HMI (see https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_DALI/Resources/zip/10466952971.zip).

This project can be extended for specific applications, or it can only be used for configuration purposes. The KL6821 is supported as standard; the user can add the KL6811.

The dialogs offer not only functions for addressing control gears and control devices, but also for writing and reading parameters.

Using the drop-down menu, the user can select whether to parameterize control gears or control devices.
Addressing of the devices and querying of the addresses that are already assigned takes place under the 
\textit{Addressing} tab, which is available for both control gears and sensors.

Parameters can be read based on selected short addresses and can be written, as far as possible.
There are also dialogs for

- Executing DALI commands
- Reading current states
- Reading and configuring group allocations
- Reading and configuring assignments within scenes
- Reading and writing the configuration of the following device types:
  - 0 (generic inputs, part 103)
  - 1 (push buttons, part 301)
  - 2 (absolute input devices, part 302)
  - 3 (occupancy sensors, part 303)
  - 4 (light sensor, part 304)
  - 6 (LED modules, part 207)

For this purpose, one page is available for reading the configuration and one for writing.

- Adjustment of the fading settings
- Reading and, if the vendor wishes, configuration of the memory banks.
- Memory bank 1 for the control gears (part 102) contains the extension (device type 50).
- The device type 51 (energy report) is available by querying the memory banks 202, 203 and 204.
- The device type 52 (diagnostics and maintenance) is available by querying the memory banks 205, 206 and 207.
- Configuration of the KL6821 (switching the power supply unit on/off, behavior on triggering of the K-Bus watchdog, behavior in case of rising or falling edges on the two digital inputs of the terminal, enablement of the process image after the inputs were actuated, deactivation of the blocking of the process image when using the digital inputs of the KL6821).
### Requirements

<table>
<thead>
<tr>
<th>Required PLC library</th>
<th>DALI Commissioning Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc3_DALI from v3.6.0.0</td>
<td>Tc3_DALI_Commissioning v3.6.0.0</td>
</tr>
</tbody>
</table>

#### 4.1.3 Event Logger

The PLC library Tc3_DALI supports the TwinCAT Event Logger for the output of messages. As a result, the developer is also shown errors without explicitly querying the error IDs of the individual instances in the PLC program.

The output of the messages can be influenced by a variable. It is located in the library Tc3_DALI as a global variable:

```plaintext
eEventTraceLevel : TcEventSeverity := TcEventSeverity.Critical;
```

Each message is assigned to a level that indicates how serious it is. The levels Info, Warning, Error or Critical are available. The global variable eEventTraceLevel defines the level from which a message is displayed in the message window.

By default, the variable is initialized so that only messages that correspond to the Critical level are output. Since most messages of the Tc3_DALI library are assigned to the Error level, these messages are not displayed. The Level Critical is rated more serious than the Level Error.

#### PLC

The following instruction, which you can insert into your PLC program, outputs all messages of all function blocks from the Tc3_DALI library. This is particularly helpful during development or commissioning.

```plaintext
Tc3_DALI.GVL.eEventTraceLevel := TcEventSeverity.Verbose;
```

Each function block also contains a local variable that overrides the global variable setting for the corresponding instance. In the online view, the output of the messages can be defined individually for each instance:

<table>
<thead>
<tr>
<th>Instance</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UiTraceLevel</td>
<td>BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>eTraceLevel</td>
<td>TCEVENTSEVERITY</td>
<td>Verbose</td>
</tr>
<tr>
<td>ipCommunication</td>
<td>I_DALICommunication</td>
<td>16#E43046C0</td>
</tr>
<tr>
<td>ipCommunicationPreset</td>
<td>I_DALICommunication</td>
<td>16#00000000</td>
</tr>
<tr>
<td>bStart</td>
<td>BOOL</td>
<td>TRUE</td>
</tr>
<tr>
<td>nAddress</td>
<td>BYTE</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Watch List

In addition to the extension of the PLC program, you can create a watch list in which the global variable eEventTraceLevel can be changed.
1. Create the watch list.

![Image of TwinCAT interface showing Watch 1 and Watch 2 options]

2. Select the global variable `eEventTraceLevel`.

![Image of TwinCAT Input Assistant with the variable eEventTraceLevel highlighted]

The default setting of the variable can be seen in the documentation area of the input assistant.

⇒ After adding the variable to the watch list you can change its value via a drop-down menu.

**Displaying the messages**

The messages are displayed in the TwinCAT Logged Events window (see chapter Event Logger).
All messages are stored in a ring buffer in the TwinCAT controller. The size of the ring buffer can be adapted in the TwinCAT options. This has the advantage that messages are saved even if the TwinCAT project is not open in Visual Studio.

In addition, the messages can also be displayed in the **Output** window.

Make sure that **Show output from** is set to **TwinCAT**.

### 4.1.4 Communication Monitor

The TwinCAT 3 Extension *TwinCAT Communication Monitor* contains a diagnostic window in which the DALI commands sent and the DALI events of the Tc3_DALI library are displayed.

**Installation with TwinCAT XAE (TCXaeShell)**

If you use the TwinCAT XAE Shell, perform the following steps to install the *TwinCAT Communication Monitor*:

- If you are using a 32-bit operating system, create the folder *TwinCAT Communication Monitor* under the following path: *C:\Program Files\Beckhoff\TcXaeShell\Common7\IDE\Extensions\Beckhoff Automation GmbH*. 

---

**Logged Events**

<table>
<thead>
<tr>
<th>Target System</th>
<th>Time Received</th>
<th>Severity Level</th>
<th>Event</th>
<th>Source Name</th>
<th>Event Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2.168.1.100</td>
<td>07.12.2020 09:52:26.385</td>
<td>Error</td>
<td>10</td>
<td>MAIN_OFF_DALI</td>
<td>Parameter &quot;Address&quot; (actual value: 73) is a short address and lies outside of the valid range (0-63).</td>
</tr>
<tr>
<td>12.2.168.1.100</td>
<td>07.12.2020 09:52:20.848</td>
<td>Error</td>
<td>262</td>
<td>MAIN_OFF_DALI</td>
<td>Parameter &quot;Address&quot; (actual value: 73) is a short address and lies outside of the valid range (0-63).</td>
</tr>
</tbody>
</table>

**Options**

*MaxLoggedEvents*: 100
*Output as Task Item*: True
For a 64-bit operating system, the folder must be created under: C:\Program Files (x86)\Beckhoff\TcXaeShell\Common7\IDE\Extensions\Beckhoff Automation GmbH.

Download the TwinCAT Communication Monitor from the following link and save the *.vsix file that it contains locally: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_DALI/Resources/zip/9260769675.zip

Rename the *.vsix file to a *.zip file.

Copy the contents of the previously renamed file to the newly created folder.

Check that the following security settings are set correctly for all files:

- Restart TwinCAT XAE (TcXaeShell).

**Installation with Visual Studio**

Download the TwinCAT Communication Monitor from the following link and save the *.vsix file that it contains locally:

https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_DALI/Resources/zip/9260769675.zip

The TwinCAT Communication Monitor is installed by double-clicking the TcCommunicationMonitorVSIX_V1.1.0.0.vsix file.

**Start the DALI TwinCAT Communication Monitor**

After the TwinCAT Communication Monitor has been installed, the DALI Monitor entry is available under the TwinCAT menu item.
After starting the DVALID Monitor, select the instance of the desired communication function block (FB_KL6811Communication \[341\] or FB_KL6821Communication \[343\]). All telegrams sent by the respective instance are then displayed in the DVALID Monitor.

In addition to the timestamp, address, DALI command and command priority, the instance path is also displayed. This indicates from where within the PLC program the DALI command was sent.

To display the telegrams, it is not necessary that the corresponding TwinCAT project has been loaded. Since the data exchange between the PLC program and the DVALID Monitor was realized via ADS, it is sufficient if the TwinCAT router was correctly configured.

### 4.1.5 Error evaluation

The interface pointer ipResultMessage of type I_TcMessage provides information about a current event (runtime message [561]). The most important methods and properties are described below:

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EqualsToEventEntryEx</td>
<td>Compares the event definition of the event with another event definition.</td>
</tr>
<tr>
<td>RequestEventText</td>
<td>Returns the text for the event.</td>
</tr>
</tbody>
</table>
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSeverity</td>
<td>TcEventSeverity</td>
<td>Returns the severity.</td>
</tr>
<tr>
<td>EventClass</td>
<td>GUID</td>
<td>Returns the unique ID of the event class.</td>
</tr>
<tr>
<td>nEventId</td>
<td>UDINT</td>
<td>Returns the ID of the event.</td>
</tr>
<tr>
<td>stEventEntry</td>
<td>TcEventEntry</td>
<td>Returns the event definition.</td>
</tr>
</tbody>
</table>

For details about the interface I_TCMMessage, see the documentation for the Tc3_EventLogger library.

Each event is unambiguously described by the structure TcEventEntry. It contains three structure elements:

- Event-Class
- Event-ID
- Event-Severity

Several events are merged into one event class. For example, the event class for DALI and the event class for EnOcean can contain the event ID 100. The complete description of an event always consists of the event class and the event ID. Ambiguities due to the same IDs within different libraries are thus avoided.

In order to make the events accessible to the PLC, TwinCAT automatically creates a structure for each event class in the type system.

This happens for all PLC libraries that are referenced in the project. The structure ST_DALIEventClass is added using the Tc3_DALI library.
This structure contains all events of the event class for the Tc3_DALI library. Each element of the structure is mapped by a variable of type \textit{TcEventEntry}. The name of the element indicates the meaning of the event.

\textbf{TYPE} \textit{ST_TcDALIEventClass}:
\begin{verbatim}
  STRUCT
    NoResponseFromDALITerminal: TcEventEntry := ( ...
    NoResponseFromDALIDevice: TcEventEntry := ( ...
    CommandBufferOverflow: TcEventEntry := ( ...
  END_STRUCT
END_TYPE
\end{verbatim}

All event classes are stored in the global variable list \textit{TC_Events}, which is also generated automatically by TwinCAT.

\textbf{VAR_GLOBAL CONSTANT}
\begin{verbatim}
  [..]
  TcRTimeEventClass: ST_TcRTimeEventClass;
  Win32EventClass: ST_Win32EventClass;
  TcDALIEventClass: ST_TcDALIEventClass;
END_VAR
\end{verbatim}

Thus all messages of all event classes are available in the PLC program.

**FAQ**

**How can I query whether an FB returns an error?**

In all libraries, the event ID for error-free execution is set to 0. Since the event ID not only outputs errors, but also notes and warnings, most function blocks have the output \textit{bError}. This output is TRUE when an error occurs.

\begin{verbatim}
IF (fbDALI.bError) THEN
... END_IF
\end{verbatim}

This output remains FALSE in the event of warnings or notes. Therefore it may be the case that the event ID (\textit{fbDALI.ipResultMessage.nEventID}) is not 0, but \textit{bError} remains FALSE.
How can I query whether a FB returns a specific event?

The method

\[
\text{ipResultMessage.EqualsToEventEntryEx(stOther\_TcEventEntry)}
\]

compares whether the event matches the transferred event \(stOther\). Since a variable of type \(TcEventEntry\) automatically exists for each event, the query for a certain event can be carried out as follows.

\[
\begin{align*}
\text{IF} & \ (\text{fbDALI.ipResultMessage.EqualsToEventEntryEx(} \\
& \quad \text{TC EVENTS.TcDALIEventClass.NoResponseFromTheDALIDevice)} \text{) THEN} \\
\end{align*}
\]

Alternatively, the Event ID can also be queried directly.

\[
\begin{align*}
\text{IF} & \ (\text{fbDALI.ipResultMessage.nEventId} = 2) \text{ THEN} \\
\end{align*}
\]

A structure of the data type \(TcEventEntry\) exists for each event. This is located inside the structure \(TcDALIEventClass\) (data type \(ST\_TcDALIEventClass\)). The meaning of the message is recognizable from the names of the individual events. The use of this structure can thus improve the readability of the PLC program:

\[
\begin{align*}
\text{IF} & \ (\text{fbDALI.ipResultMessage.nEventId} = \\
& \quad \text{TC EVENTS.TcDALIEventClass.NoResponseFromTheDALIDevice.nEventId}) \text{ then} \\
\end{align*}
\]

For a list of all events that represent an error, see Runtime messages [561].

How can I query the text of an event?

The text that is output in the TwinCAT Logged Events window can also be queried in the PLC program using the method

\[
\text{ipResultMessage.RequestEventText(nLangId\ DINT, sResult\ REFERENCE\ TO\ STRING, nResultSize\ UDINT)}
\]

If an event is present, the following sample returns the event text:

\[
\begin{align*}
s\text{Txt} & \quad : \quad \text{STRING(255)}; \\
s\text{EventText} & \quad : \quad \text{STRING(255)}; \\
\text{IF} & \ (\text{fbDALI.ipResultMessage.RequestEventText(1031, sTxt, SIZEOF(sTxt))}) \text{ THEN} \\
& \quad \text{UTF8\_TO\_STRING(ADR(sEventText), ADR(sTxt), SIZEOF(sTxt));} \\
\end{align*}
\]

The first parameter specifies the language in which the text is to be read. The Tc3\_DALI library contains all texts in English (1033) and German (1031).

4.2 Tc2\_DALI and Tc3\_DALI in the same project

The Tc2\_DALI library can be used for porting existing TwinCat 2 projects to TwinCAT 3. At source code level, the TwinCAT 2 DALI library is compatible with the TwinCAT 3 Tc2\_DALI library, which significantly simplifies the transfer of TwinCAT 2 projects to TwinCAT 3.

In contrast, the TwinCAT 3 library Tc3\_DALI is a completely new development, which has been optimized for TwinCAT 3 and adapted to DALI-2. In this way it was possible to improve the performance and to simplify handling of the function blocks further.

For new projects the Tc3\_DALI library should therefore be used if possible.

This can lead to a situation where the Tc2\_DALI library is used in a TwinCAT 3 project, this project is subsequently extended and the Tc3\_DALI library is to be used for this purpose.
The following section illustrates how a KL6821 with the Tc2_DALI library and a further KL6821 with the Tc3_DALI library are operated in a TwinCAT 3 project.

The function blocks of the Tc2_DALI and Tc3_DALI libraries cannot be mixed on the same Bus Terminal. Each library must have exclusive access to a KL6821 or KL6811.

Add both libraries to the TwinCAT 3 project:

```
VAR_GLOBAL
  // Tc2_DALI
  stKL6821InDataTc2 AT %I* : Tc2_DALI.ST_KL6821InData;
  stKL6821OutDataTc2 AT %Q* : Tc2_DALI.ST_KL6821OutData;
  stCommandBufferKL6821 : Tc2_DALI.ST_DALIV2CommandBuffer;
  // Tc3_DALI
  stKL6821InDataTc3 AT %I* : Tc3_DALI.ST_KL6821InData;
  stKL6821OutDataTc3 AT %Q* : Tc3_DALI.ST_KL6821OutData;
END_VAR
```

Since the structures for linking the Bus Terminal have the same name in both libraries, the namespace must be prefixed. Although the structures have the same name in both libraries, they have different data types within the TwinCAT 3 type system. The namespace ensures that the appropriate data type from the respective library is used. For better readability, the namespace was also specified for the structure for the command buffer (ST_DALIV2CommandBuffer) from the Tc2_DALI library. Strictly speaking this would not be necessary at this point, because the identifier ST_DALIV2CommandBuffer only occurs in the Tc2_DALI library.

The namespace of a library is displayed in the Properties window.

By convention, the namespace of the Beckhoff library is the same as the name of the library.
Next, the function block that is called by the communication task is created. In this function block, an instance of FB_KL6821Communication is called for each terminal, among other things. The namespace must also be specified here, since each library contains a function block with the name FB_KL6821Communication.

```
PROGRAM Communication
VAR
  // Tc2_DALI
  fbKL6821CommunicationTc2 : Tc2_DALI.FB_KL6821Communication;

  // Tc3_DALI
  fbKL6821CommunicationTc3 : Tc3_DALI.FB_KL6821Communication;
END_VAR

// Tc2_DALI
fbKL6821CommunicationTc2(stInData := GVL.stKL6821InDataTc2,
  stOutData := GVL.stKL6821OutDataTc2,
  stCommandBuffer := GVL.stCommandBufferKL6821);

// Tc3_DALI
fbKL6821CommunicationTc3(stInData := GVL.stKL6821InDataTc3,
  stOutData := GVL.stKL6821OutDataTc3);
```

When calling the respective function blocks for sending the DALI commands or for receiving the events, in most cases it is not necessary to specify the namespace. In the Tc2_DALI library all elements have the prefix `DALIV2` while in the Tc3_DALI the prefix `DALI` is used.

You can download the sample project from here: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_DALI/Resources/zip/9843858059.zip

### 4.3 Creating an application function block

The following section illustrates how to create an application function block based on the library Tc3_DALI. The example describes a simple function block that controls control gears via short address, group address or broadcast.

The TwinCAT project with the example can be found https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_DALI/Resources/zip/18014408379327115.zip:

```
FB_DALIApplicationExample

- bInitialize
- bBusy
- nAddress
- eAddressType
- bRecallMaxLevel
- bRecallMinLevel
- bOff
- nReferenceDeviceAddress
- tCycleActualLevel

A positive edge at one of the three inputs sends the DALI command RECALL MAX LEVEL, RECALL MIN LEVEL or OFF.

**Parameterization of the DALI control gears**

The function block can initialize the variables `maxLevel`, `minLevel` and `fadeTime` of DALI control gears. Corresponding properties are available for this purpose. Their values are written to the DALI control gears when a positive edge is detected at `bInitialize`. During this time the output `bInitialize` is TRUE.

**DALI short address for reference device (nReferenceDeviceAddress)**

The parameter `nReferenceDeviceAddress` is used for reading the output value of a reference DALI control gear cyclically in the background (`tCycleActualLevel > 0 s`). The output value is available at the output `nActualLevel`. 
No reference DALI control gear is required if a single DALI control gear is to be controlled with the function block \((eAddressType = E\_DALIAddressType\_Short)\). In this case, the output value of the individual control gear is determined via the individual address \(nAddress\).

If a DALI group is addressed with the function block \((eAddressType = E\_DALIAddressType\_Group)\), \(nReferenceDeviceAddress\) must always contain the short address of a DALI control gear, which must be located in the addressed DALI group.

Even with a broadcast \((eAddressType = E\_DALIAddressType\_Broadcast)\), a DALI control gear with a corresponding short address must be located on the DALI line.

**Transfer of the reference to the communication block**

The method \(FB\_init\) transfers the parameter \(ipDALICommunication\) of type \(I\_DALICommunication\) to the function block. The interface \(I\_DALICommunication\) is included in the Tc3\_DALI library and is implemented by all communication blocks (FB\_KL6811\_Communication and FB\_KL6821\_Communication). All DALI command function blocks (e.g. FB\_DALI102\_Off) communicate with the DALI communication block via this interface. In the method \(FB\_init\) the interface pointer \(ipDALICommunication\) is assigned to the variable \(_ipDALICommunication\) of the function block.

To ensure that the reference can be changed at runtime, the interface pointer is also passed to the application function block via the property \(ipDALICommunication\). Internally, the property is stored in the variable \(_ipDALICommunication\).

In the application function block it must be ensured that this interface pointer is passed on to all DALI command function blocks. The internal method \(SetCommunication\) is available for this purpose. In this method the interface pointer is passed to all DALI command function blocks in the function block. The property \(ipDALICommunication\) is used in the command function blocks for this purpose. This property is used to reassign the reference to the DALI communication block at runtime.

\(SetCommunication\) is called in the first PLC cycle, each time a positive edge is detected at one of the inputs \(bInitialize\), \(bRecallMaxLevel\), \(bRecallMinLevel\) or \(bOff\), and when the current output value is to be read. This ensures that the interface pointer is only passed on if no DALI command function block is currently being executed.

**Checking the parameters**

All parameters are checked by the internal method \(CheckParameters\). In the event of an error the method returns the error Id, or 0 if there is no error. The Ids of the runtime messages [561] can be used here:

```plaintext
IF (THIS^.ipDALICommunication = 0) THEN
    CheckParameters := TC\_EVENTS.TcDALIEventClass.TheInterfaceToTheCommunicationBufferIsNotInitialized.nEventId;
END_IF
```

If required, you can also use your own error Ids, starting at 1000:

```plaintext
IF (<check your parameter>) THEN
    CheckParameters := 1000;
END_IF
```

Similar to \(SetCommunication\), \(CheckParameters\) is also called in the first PLC cycle. This call also occurs if a positive edge was detected at the inputs \(bInitialize\), \(bRecallMaxLevel\), \(bRecallMinLevel\) or \(bOff\).

Alternatively, you could call the method every time the value of a parameter has changed.

**Structure of the function block**

In the upper part of the function block the positive edges of the inputs \(bInitialize\), \(bRecallMaxLevel\), \(bRecallMinLevel\) or \(bOff\) are queried.

The middle part contains a step sequence in which the individual actions are carried out. If a positive edge is detected at one of the inputs, the corresponding step of the step sequence is triggered. The DALI command \(QUERY\ \text{ACTUAL\ LEVEL}\) is sent once the DALI commands \(RECALL\ \text{MAX\ LEVEL}\) (step 200) or \(RECALL\ \text{MIN\ LEVEL}\) (step 300) have been executed. This reads the new output value from the DALI reference device and outputs it at the output \(nActualLevel\). The variables \(maxLevel\), \(minLevel\) and \(fadeTime\) are written from step 10000.
In the lower part, the output value is read cyclically from the DALI reference device and output at the output \( n_{\text{ActualLevel}} \).

**Comments**

The step sequence can only be set to a new action if no action is currently being executed (\( n_{\text{Step}} = 0 \)).

The number of DALI commands should be minimized. For example, after the DALI commands RECALL MAX LEVEL or RECALL MIN LEVEL, the current output value is only read out if during the next 1000 ms the output value is not already read out by cyclic querying in the lower part of the function block:

```plaintext
IF (((tonCycleActualLevel.PT - tonCycleActualLevel.ET) > T#1S) OR (tCycleActualLevel = T#0S)) THEN
  nStep := 9000;
ELSE
  bBusy := FALSE;
  nStep := 0;
END_IF
```