Documentation of the library
WagoAppAppLED
Release 1.7.2.0
1 Description 1

2 20 Program Organization Units 3
   2.1 10 Compact ........................................ 3
       2.1.1 FbAppLED_SetBlink (FB) .......................... 3
       2.1.2 FbAppLED_SetFlash (FB) .......................... 4
       2.1.3 FbAppLED_SetOn (FB) .............................. 6
       2.1.4 FbAppLED_SetSequence (FB) ......................... 7
       2.1.5 FbAppLED_SetStatic (FB) ......................... 8
   2.2 30 Base ............................................. 9
       2.2.1 FbAppLED (FB) .................................... 9

10 Main Functions ........................................ 11
   FbAppLED.SetBlink (METH) ............................... 11
   FbAppLED.SetFlash (METH) ............................... 12
   FbAppLED.SetOff (METH) ................................ 13
   FbAppLED.SetOn (METH) ................................... 14
   FbAppLED.SetStatic (METH) .............................. 14

20 Sequence ............................................... 15
   FbAppLED.SetSequence (METH) ........................... 15
   FbAppLED.UpdateSequence (METH) ......................... 16

3 29 Types ............................................... 17
   3.1 eLedMode (ENUM) ...................................... 17
   3.2 typLedSequenceStep (STRUCT) ......................... 17

4 80 Status ............................................... 18
   4.1 ResultItems (GVL) ..................................... 18

5 VersionHistory (GVL) .................................... 20

6 Library Reference ....................................... 21
Description

This document is automatically generated. Because of this, the chapter 30 Visualization is not shown in this document. If you are interested in getting to know more about visualization, we refer to the library manager of e!Cockpit.

Subject to Changes

WAGO Kontakttechnik GmbH Co. KG reserves the right to provide for any alterations or modifications. WAGO Kontakttechnik GmbH Co. KG owns all rights arising from the granting of patents or from the legal protection of utility patents. Third-party products are always mentioned without any reference to patent rights. Thus, the existence of such rights cannot be excluded.

Personnel Qualification

All tasks that are carried out with libraries made for the e!COCKPIT software must only be performed by qualified electrical specialists instructed in PLC programming according to IEC 61131-3.

All tasks that have an effect on the properties or the behavior of automation hardware or software products must only be performed by qualified employees with a thorough knowledge of handling the products concerned.

Intended Use of e!COCKPIT Libraries

Libraries created for the e!COCKPIT software are used to simplify the development of application projects in the IEC 61131-3 programming languages.

For automation tasks, WAGO offers programmable logic controllers in a wide variety of performance classes. In combination with a wide range of I/O modules, the controllers can process standard types of field signals. Controllers can be implemented centrally or in decentralized configurations. The controllers offer interfaces for the most commonly used fieldbuses for use in decentralized configurations. Fieldbus independent I/O modules are then linked via fieldbus couplers. WAGO controllers offer a runtime environment for user programs called e!RUNTIME. Software projects for implementation in e!RUNTIME environments can be created in e!COCKPIT. The programming environment in e!COCKPIT is based on the established CODESYS 3 industrial standard. Users with a previous knowledge of CODESYS 3 will thus find this environment largely familiar. The following programming languages of the IEC 61131-3 standard are available:

- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Sequential Function Chart (SFC)
- Continuous Function Chart (CFC)

The individual programming languages can also be combined as required during the development of the software. A portfolio of prepared libraries can be accessed for many frequently used functions in order to make software development more efficient. This document provides an overview of the WagoAppAppLED that WAGO offers for e!COCKPIT.
This library provides an interface for the LEDs which are located on several WAGO PLCs and which may be controlled by the user.¹

Further library information are summerized here:

Company WAGO
Title WagoAppAppLED
Version 1.7.2.0
Categories WAGO LayerView|App; Application; WAGO FunctionalView|Device|Controller; WAGO Internal|Feature|Common|AppLED
Author WAGO / u013972
Placeholder WagoAppAppLED
20 Program Organization Units

2.1 10 Compact

2.1.1 FbAppLED_SetBlink (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
<th>Inherited from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>oStatus</td>
<td>FbResult</td>
<td>Status object (see WagoSysErrorBase).</td>
<td>FbBehaviour-Model_oStatus_Base</td>
</tr>
<tr>
<td>Input</td>
<td>xEnable</td>
<td>BOOL</td>
<td>Enables the operation.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td>Output</td>
<td>xError</td>
<td>BOOL</td>
<td>Indicates that an Error has occurred.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td></td>
<td>xValid</td>
<td>BOOL</td>
<td>Indicates that Data is valid.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td></td>
<td>xBusy</td>
<td>BOOL</td>
<td>Indicates that the FB is working.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td>Input</td>
<td>eLedID</td>
<td>WagoTypesAp-</td>
<td>The LED which is handled here</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pLED.eLedID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tTime1</td>
<td>TIME</td>
<td>On-time for first state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tTime2</td>
<td>TIME</td>
<td>On-time for second state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>eLedColor1</td>
<td>WagoTypesAp-</td>
<td>Color in first state</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pLED.eLedColor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>eLedColor2</td>
<td>WagoTypesAp-</td>
<td>Color in second state</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pLED.eLedColor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function

The function block FbAppLED_SetBlink_cpt() makes the LED toggle periodically between two colors.

Graphical Illustration

![Function Block Diagram](attachment:image)
Function description

If \( xEnable \) true, then \( eColor1 \) is displayed for the duration \( tTime1 \). After that on-time has elapsed, the second color (\( eColor2 \)) appears for a duration of \( tTime2 \). After that second time, this process repeats with \( eColor1 \). A falling edge at \( xEnable \) stop the execution and switch off th LED, for a new execution or a parameter change a rising edge at \( xEnable \) is necessary.

The status output \( xBusy \) is set, while \( xEnable \) is set and the functionblock is still in progress. The status output \( xValid \) indicate a successful operation, it will be rest with a falling edge at \( xEnable \). The status output \( xError \) indicate a problem in the execution of the functionblock, it will be reset one cycle after \( xBusy \) was reset.

On error, no change of the LED state takes place.

<table>
<thead>
<tr>
<th>result codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>ENOENT</td>
<td>The desired LED-ID does not exist on this PLC</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>invalid parameters</td>
</tr>
</tbody>
</table>

Note: For the PFCX00-System are only the Colors Green, Red and Yellow supported.

### 2.1.2 FbAppLED_SetFlash (FB)

Interface variables
### Function

The function block `FbAppLED_SetFlash_cpt()` displays a color for a short time and displays a second color statically afterwards.

#### Graphical Illustration

![Function Block Diagram](image)

#### Function description

If `xEnable` true, then the color of this ‘flash’ is given by `eColor1` and its duration is given by `tFlashTime`. After `tFlashTime` has elapsed, the LED displays the other color `eColor2` (which may be a visible color as well as simply off). A falling edge at `xEnable` stop the execution and switch off th LED, for a new execution or a parameter change a rising edge at `xEnable` is necessary.

The status output `xBusy` is set, while `xEnable` is set and the functionblock is still in progress. The status output `xValid` indicate a successful operation, it will be rest with a falling edge at `xEnable`. The status output `xError` indicate a problem in the execution of the functionblock, it will be reset one cycle after `xBusy` was reset.
While the first phase of the flash is in progress, the \texttt{xBusy}-Output of the FB is \texttt{TRUE}.

When applying this function while the LED is still ‘flash’ing, the flash time will be retriggered.

When the flash time is expired, the LED transits into the static mode.

\textbf{Note:} Some legacy libraries require the LED to be driven into in certain other operating modes (namely ‘static’)) before the LED could be driven to flash mode. This FB, however, has no such restriction.

On error, no change of the LED state takes place.

\begin{table}[h]
\centering
\begin{tabular}{|c|p{9cm}|}
\hline
\textbf{result codes} & \textbf{Description} \\
\hline
0 & \texttt{Success} \\
EBADF & The Fb is not in the Open (=Enabled) state \\
EBADR & Requested Color is not supported \\
EINV AL & Invalid parameters \\
\hline
\end{tabular}
\end{table}

\textbf{Note:} For the PFCX00-System are only the Colors Green, Red and Yello supported.

\subsection{2.1.3 FbAppLED\_SetOn (FB)}

\textbf{Interface variables}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|p{12cm}|}
\hline
\textbf{Scope} & \textbf{Name} & \textbf{Type} & \textbf{Comment} & \textbf{Inherited from} \\
\hline
Output & oStatus & FbResult & Status object (see WagoSysErrorBase). & FbBehaviour-Model\_oStatus\_Base \\
Input & xEnable & BOOL & Enables the operation. & FbBehaviour-Model\_WagoAppEnable \\
Output & xError & BOOL & Indicates that an Error has occurred. & FbBehaviour-Model\_WagoAppEnable \\
 & xValid & BOOL & Indicates that Data is valid. & FbBehaviour-Model\_WagoAppEnable \\
 & xBusy & BOOL & Indicates that the FB is working. & FbBehaviour-Model\_WagoAppEnable \\
Input & eLedID & WagoTypesAp-pLED\_eLedID & The LED which is handled here \\
\hline
\end{tabular}
\end{table}

\textbf{Function}

The functionblock \texttt{FbAppLED\_SetOn\_cpt()} switches the LED to a device specific color which is guaranteed to be visible.

\textbf{Graphical Illustration}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{functionblock.jpg}
\end{figure}

\textbf{Function Description}

This functionblock is similar to \texttt{FbAppLED\_SetStatic\_cpt()}, but it guarantees that the LED will turn on in some default color. When using \texttt{FbAppLED\_SetStatic\_cpt()} in contrast, the user may select the color,
but the LED will not turn on if he selects a color which cannot be displayed (e.g. blue on a red-green-LED). A falling edge at \texttt{xEnable} stop the execution and switch off the LED, for a new execution or a parameter change a rising edge at \texttt{xEnable} is necessary.

The status output \texttt{xBusy} is set, while \texttt{xEnable} is set and the functionblock is still in progress. The status output \texttt{xValid} indicate a successful operation, it will be rest with a falling edge at \texttt{xEnable}. The status output \texttt{xError} indicate a problem in the execution of the functionblock, it will be reset one cycle after \texttt{xBusy} was reset.

\textbf{Note:} Which color is used as default color depends on the target hardware and on the type of the LED. It is hardcoded in the target depending internal libraries and cannot be changed by the user. It is ensured that the default color is visible for the addressed target hardware.

On error, no change of the LED state takes place.

<table>
<thead>
<tr>
<th>\textbf{result codes}</th>
<th>\textbf{Comment}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBAFD</td>
<td>The FB is not in the Open (=Enabled) state</td>
</tr>
</tbody>
</table>

\textbf{Note:} For the PFCX00-System are only the Colors Green, Red and Yello supported.

### 2.1.4 \texttt{FbAppLED_SetSequence} (FB)

The functionblock \texttt{FbAppLED_SetSequence} starts the display of a blink code for indicating a specific status or error condition.

#### Graphical Illustration

\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Scope} & \textbf{Name} & \textbf{Type} & \textbf{Comment} & \textbf{Inherited from} \\
\hline
Output & oStatus & FbResult & Status object (see WagoSysErrorBase) & FbBehaviour-Model\_oStatus\_Base \\
\hline
Input & xEnable & BOOL & Enables the operation. & FbBehaviour-Model\_WagoAppEnable \\
\hline
Output & xError & BOOL & Indicates that an Error has occurred. & FbBehaviour-Model\_WagoAppEnable \\
& xValid & BOOL & Indicates that Data is valid. & FbBehaviour-Model\_WagoAppEnable \\
& xBusy & BOOL & Indicates that the FB is working. & FbBehaviour-Model\_WagoAppEnable \\
\hline
Input & eLedID & WagoTypesAppLED\_eLedID & The LED which is handled here & \\
& pSequence & POINTER TO ARRAY [0..0] OF typLedSequenceStep & List of steps of the sequence & \\
& udiSequenceSize & UDINT & Number of steps & \\
\hline
\end{tabular}
Function description

Each pulse of the sequence has an individually settable duration and color. If off-phases are required between the pulses, these off-phases have to be represented by an individual sequence step with color **off**.

The input `pSequence` points to an array of sequence steps (**typLedSequenceStep** (**STRUCT**)). This array has a variable length which is denoted by `udiSequenceSize`. When a sequence step carries a duration of 0, this also denotes the end of the sequence, even if `udiSequenceSize` indicates more steps.

Once initiated, the sequence cannot be altered - but it can be stopped or restarted instead. While the sequence is in progress, the `xBusy`-Output of the FB is **TRUE**.

After the sequence has terminated, the LED transits to the **static** mode and displays the color of the last sequence step (i.e. the one with **T#0s**, when the sequence contains such a step).

A falling edge at `xEnable` stop the execution and switch off the LED, for a new execution or a parameter change a rising edge at `xEnable` is necessary.

The status output `xBusy` is set, while `xEnable` is set and the functionblock is still in progress. The status output `xValid` indicate a successful operation, it will be rest with a falling edge at `xEnable`. The status output `xError` indicate a problem in the execution of the functionblock, it will be reset one cycle after `xBusy` was reset.

<table>
<thead>
<tr>
<th>result codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The FB is not in the Open (=Enabled) state</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Illegal sequence description or no sequence given</td>
</tr>
</tbody>
</table>

**Note:** For the PFCX00-System are only the Colors Green, Red and Yello supported.

### 2.1.5 FbAppLED_SetStatic (FB)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
<th>Inherited from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>oStatus</td>
<td>FbResult</td>
<td>Status object (see WagoSysErrorBase).</td>
<td>FbBehaviour-Model_oStatus_Base</td>
</tr>
<tr>
<td>Input</td>
<td>xEnable</td>
<td>BOOL</td>
<td>Enables the operation.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td>Output</td>
<td>xError</td>
<td>BOOL</td>
<td>Indicates that an Error has occurred.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td></td>
<td>xValid</td>
<td>BOOL</td>
<td>Indicates that Data is valid.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td></td>
<td>xBusy</td>
<td>BOOL</td>
<td>Indicates that the FB is working.</td>
<td>FbBehaviour-Model_WagoAppEnable</td>
</tr>
<tr>
<td>Input</td>
<td>eLedID</td>
<td>WagoTypesApp-pLED.eLedID</td>
<td>The LED which is handled here</td>
<td></td>
</tr>
<tr>
<td></td>
<td>eLedColor</td>
<td>WagoTypesApp-pLED.eLedColor</td>
<td>Desired color of the LED (or ‘off’).</td>
<td></td>
</tr>
</tbody>
</table>
Function
The function block FbAppLED_SetStatic_cpt() makes the LED continuously display a given color or switches the LED off.

**Graphical Illustration**

![Function Block Diagram](image)

**Function Description**
If the requested color is not supported by the hardware, this method fails with EBADR (‘Bad Request’); A falling edge at xEnable stops the execution and switches off the LED, for a new execution or a parameter change a rising edge at xEnable is necessary.

The status output xBusy is set, while xEnable is set and the function block is still in progress. The status output xValid indicate a successful operation, it will be reset with a falling edge at xEnable. The status output xError indicate a problem in the execution of the function block, it will be reset one cycle after xBusy was reset.

<table>
<thead>
<tr>
<th>result codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The Fb is not in the Open (=Enabled) state</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>invalid arguments</td>
</tr>
</tbody>
</table>

**Note:** For the PFCX00-System are only the Colors Green, Red and Yellow supported.

### 2.2 30 Base

#### 2.2.1 FbAppLED (FB)

**Interface variables**
### Function

The Function Blocks \( \text{FbAppLED} \) provides access to one of the App-LEDs of the controller in various operating modes.

### Graphical Illustration

![Function Block Diagram](function_block.png)

### Function Description

This FB implements the behaviour model WagoChannel in order to coordinate access to the LEDs from different places within the application program. A \( \text{TRUE} \) state at the input \( x\text{Open} \) opens the access to the LED and a \( \text{FALSE} \) state closes it again. (For a general description of this model, please refer to \( \text{FbBehaviourModel}_\text{WagoChannel} \)).

Before using a App-LED, the ID of the App LED has to be passed to the input \( \text{FbAppLED.eID} \). While using a App-LED, the \( x\text{Open}\)-Input has to be kept \( \text{TRUE} \). If \( \text{eID} \) denotes an LED which does not exist on the given hardware, the result code \( \text{ENOENT} \) and \( x\text{Error}=\text{TRUE} \) appears at the status output.

### Note:

If the desired LED-ID does not exist on the PLC or if a general error occurs, the FB signals its operational state with its output \( x\text{IsOpen} \).

The operation mode of an LED can be altered anytime the FB is open. If this change of operation mode is likely to give misleading LED signals to the user (e.g. because a sequence or a flash is in progress) this situation is indicated with the output \( x\text{Busy} \). Nevertheless, this is a just warning, which may be overridden by the application.

If \( x\text{Open} \) returns to \( \text{FALSE} \) again, the access to the LED is cut. Please note that the state of the LED itself (on or off) remains unchanged.
protClose which are inherited from the base model WagoAppChannel.

**Example**

```plaintext
VAR
    MyLED : FbAppLED;  // the instance for controlling an LED
    MyOtherLED : FbAppLED;  // another LED
END_VAR

MyLED(eID:=eLedID.AppLED_5, xOpen:=TRUE);  // cyclic operation, handling AppLED #5
MyLED.SetStatic(eLedColor.Red);  // switch it on
MyLED.SetStatic(eLedColor.Off);  // switch it off
MyLED.SetFlash(T#1s,eLedColor.Red, eLedColor.Green);  // initiate a 1s-pulse
IF NOT MyLED.xBusy THEN ...  // wait for the flash pulse to terminate
```

### 10 Main Functions

Most prominent methods of this FB

**FbAppLED.SetBlink (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>SetBlink</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>tTime1</td>
<td>TIME</td>
<td>On-time for first state</td>
</tr>
<tr>
<td></td>
<td>tTime2</td>
<td>TIME</td>
<td>On-time for second state</td>
</tr>
<tr>
<td></td>
<td>eColor1</td>
<td>WagoTypesAppLED.eLedColor</td>
<td>Color in first state</td>
</tr>
<tr>
<td></td>
<td>eColor2</td>
<td>WagoTypesAppLED.eLedColor</td>
<td>Color in second state</td>
</tr>
</tbody>
</table>

**Function**

The method `SetBlink()` makes the LED toggle periodically between two colors.

**Graphical Illustration**

```
Method
  FbAppLED.SetBlink
  tTime1  TIME   eResultCode
  tTime2  TIME
  eColor1 eLedColor
  eColor2 eLedColor
```

**Function description**

In the beginning, `eColor1` is displayed for the duration `tTime1`. After that on-time has elapsed, the second color (`eColor2`) appears for a duration of `tTime2`. After that second time, this process repeats with `eColor1`.  

2.2. 30 Base
On error, no change of the LED state takes place.

<table>
<thead>
<tr>
<th>Result Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The Fb is not in the Open (=Enabled) state</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Invalid parameters</td>
</tr>
</tbody>
</table>

Note: For the PFCX00-System are only the Colors Green, Red and Yello supported.

FbAppLED.SetFlash (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>SetFlash</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>tFlashTime</td>
<td>TIME</td>
<td>On-time for first state</td>
</tr>
<tr>
<td></td>
<td>eColor1</td>
<td>WagoTypesAppLED.eLedColor</td>
<td>Color in first state</td>
</tr>
<tr>
<td></td>
<td>eColor2</td>
<td>WagoTypesAppLED.eLedColor</td>
<td>Color in succeeding state</td>
</tr>
</tbody>
</table>

Function

The method SetFlash() displays a color for a short time and displays a second color statically afterwards.

Graphical Illustration

Function description

The color of this ‘flash’ is given by eColor1 and its duration is given by tFlashTime. After tFlashTime has elapsed, the LED displays the other color eColor2 (which may be a visible color as well as simply off).
While the first phase of the flash is in progress, the \texttt{xBusy}-Output of the FB is \texttt{TRUE}.

When applying this function while the LED is still ‘flash’ing, the flash time will be retriggered.

When the flash time is expired, the LED transits into the \texttt{static} mode.

\textbf{Note:} Some legacy libraries require the LED to be driven into certain other operating modes (namely ‘static’) before the LED could be driven to flash mode. This FB, however, has no such restriction.

On error, no change of the LED state takes place.

<table>
<thead>
<tr>
<th>result codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The Fb is not in the Open (=Enabled) state</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Invalid parameters</td>
</tr>
</tbody>
</table>

\textbf{Note:} For the PFCX00-System are only the Colors Green, Red and Yello supported.

\textbf{FbAppLED.SetOff (METH)}

\textbf{Interface variables}

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>SetOff</td>
<td>WagoTypes.eResultCode</td>
</tr>
</tbody>
</table>

\textbf{Function}

The method \texttt{SetOff()} sets the LED to a static \texttt{off} operating state. In case of any errors, no change of the LED state takes place.

\textbf{Graphical Illustration}

\textbf{Function Description}

<table>
<thead>
<tr>
<th>result codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The Fb is not in the Open (=Enabled) state</td>
</tr>
</tbody>
</table>
Note: For the PFCX00-System are only the Colors Green, Red and Yello supported.

FbAppLED.SetOn (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>SetOn</td>
<td>WagoTypes.eResultCode</td>
</tr>
</tbody>
</table>

Function

The method `SetOn()` switches the LED to a device specific color which is guaranteed to be visible.

Graphical Illustration

Function Description

This method is similar to `setStatic()`, but it guarantees that the LED will turn on in some default color. When using `setStatic()` in contrast, the user may select the color, but the LED will not turn on if he selects a color which cannot be displayed (e.g. blue on a red-green-LED).

Note: Which color is used as default color depends on the target hardware and on the type of the LED. It is hardcoded in the target depending internal libraries and cannot be changed by the user. It is ensured that the default color is visible for the addressed target hardware.

On error, no change of the LED state takes place.

result codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The FB is not in the Open (=Enabled) state</td>
</tr>
</tbody>
</table>

Note: For the PFCX00-System are only the Colors Green, Red and Yello supported.

FbAppLED.SetStatic (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>SetStatic</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>eColor</td>
<td>WagoTypesAppLED.eLedColor</td>
<td>Desired color of the LED (or ‘off’).</td>
</tr>
</tbody>
</table>

Function

The method `SetStatic()` makes the LED continuously display a given color or switches the LED off.

Graphical Illustration
Function Description

If the requested color is not supported by the hardware, this method fails with EBADR (‘Bad Request’);

On error, no change of the LED state takes place.

<table>
<thead>
<tr>
<th>result codes</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The Fb is not in the Open (=Enabled) state</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>invalid arguments</td>
</tr>
</tbody>
</table>

Note: For the PFCX00-System are only the Colors Green, Red and Yello supported.

20 Sequence

Methods related to generating sequences

FbAppLED.SetSequence (METH)

Interface variables

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return</td>
<td>SetSequence</td>
<td>WagoTypes.eResultCode</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>pSequence</td>
<td>POINTER TO ARRAY [0..0] OF</td>
<td>List of steps of the sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>typLedSequenceStep</td>
<td></td>
</tr>
<tr>
<td></td>
<td>udiSequence-</td>
<td>UDINT</td>
<td>Number of steps</td>
</tr>
<tr>
<td></td>
<td>Size</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function

The method SetSequence() starts the display of a blink code for indicating a specific status or error condition.

Graphical Illustration

Function description

Each pulse of the sequence has an individually settable duration and color. If off-phases are required between the pulses, these off-phases have to be represented by an individual sequence step with color off.

The input pSequence points to an array of sequence steps (typLedSequenceStep (STRUCT)). This array has a variable length which is denoted by udiSequenceSize. When a sequence step carries a duration of 0, this also denotes the end of the sequence, even if udiSequenceSize indicates more steps.

Once initiated, the sequence cannot be altered - but it can be stopped or restarted instead. While the sequence is in progress, the xBusy-Output of the FB is TRUE.
After the sequence has terminated, the LED transits to the static mode and displays the color of the last sequence step (i.e. the one with \(T\neq0\)s, when the sequence contains such a step).

<table>
<thead>
<tr>
<th>result codes</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>EBADF</td>
<td>The Fb is not in the Open (=Enabled) state</td>
</tr>
<tr>
<td>EBADR</td>
<td>Requested Color is not supported</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Illegal sequence description or no sequence given</td>
</tr>
</tbody>
</table>

**Note:** For the PFCX00-System are only the Colors Green, Red and Yellow supported.

**FbAppLED.UpdateSequence (METH)**

**Interface variables**

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>pSequence</td>
<td>POINTER TO ARRAY [0..0] OF typLedSequenceStep</td>
<td>List of steps of the sequence</td>
</tr>
<tr>
<td></td>
<td>xAbort</td>
<td>BOOL</td>
<td>TRUE aborts the sequence</td>
</tr>
</tbody>
</table>

**Function**

The method `UpdateSequence()` assigns a new `typLedSequenceStep`-pointer to the `FbAppLED` while the blink sequence is still in progress.

**Graphical Illustration**

**Function description**

If a null pointer is given for the new sequence address, the sequence will be aborted.

This method does not fail and thus does not return any result code. It may also be called while no sequence is in progress, but it has no immediate effect then until the new sequence starts.

**Note:** This method must be called at Online-Change by the embedding FB if it uses blink sequences, because at Online-Change absolute locations may change and pointers have to be updated afterwards.

**Note:** This Method may be called even if the FB is not open, because it does not initiate any action.

**Note:** For the PFCX00-System are only the Colors Green, Red and Yellow supported.
3.1 eLedMode (ENUM)

<table>
<thead>
<tr>
<th>Name</th>
<th>Initial</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail</td>
<td>0</td>
<td>Unknown state or failure</td>
</tr>
<tr>
<td>Off</td>
<td>1</td>
<td>Static Off</td>
</tr>
<tr>
<td>Static</td>
<td>2</td>
<td>Static any non-off-color</td>
</tr>
<tr>
<td>Blink</td>
<td>3</td>
<td>Blink mode</td>
</tr>
<tr>
<td>Sequence</td>
<td>4</td>
<td>Display of an error sequence</td>
</tr>
<tr>
<td>Flash</td>
<td>5</td>
<td>A single shot</td>
</tr>
</tbody>
</table>

**Function**

This enumeration reflects the operating mode, which an LED is in according to the `SetXxxx()`-method.

3.2 typLedSequenceStep (STRUCT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>eColor1</td>
<td>WagoTypesAppLED.eLedColor</td>
<td>Color of the step</td>
</tr>
<tr>
<td>tDuration</td>
<td>TIME</td>
<td>Duration of the step. 0 = end of sequence.</td>
</tr>
</tbody>
</table>

**Function**

One step of a sequence of blink codes.

**Context:**

An array of variables of this type will be passed to the FB when a sequence of flashes is started (`SetSequence`). The length of the sequence will be passed in an extra UDINT. As an additional feature, a duration time of T#0s in a `typLedSequenceStep` will also denote the end of the sequence.
### 4.1 ResultItems (GVL)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Type</th>
<th>Initial</th>
</tr>
</thead>
</table>

**Function**

This is the summary of all standard result items specific for this library.
This is a general mapping of result codes to short standard texts which are appropriate to the usage of these codes in this library.

Typically, each unit (function, method, or function block) in this library uses only a subset of these codes. Please, refer to the documentation of the specific unit for the set of codes which is actually used and for a detailed explanation of the meaning of a result code in the specific context.
### VersionHistory (GVL)

<table>
<thead>
<tr>
<th>Name</th>
<th>Info</th>
<th>Type</th>
<th>date</th>
<th>version</th>
<th>author</th>
<th>change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties: free</td>
<td>WagoSysVersion.ProjectInfo</td>
<td></td>
<td>08.01.2019</td>
<td>1.7.2.0</td>
<td>WAGO / u015842</td>
<td>free placeholder added</td>
</tr>
<tr>
<td>placeholder added</td>
<td></td>
<td></td>
<td>06.02.2018</td>
<td>1.7.1.1</td>
<td>WAGO / u013972</td>
<td>Change in the Documentation</td>
</tr>
<tr>
<td>WAT19585 - Behaviour of the status-outputs</td>
<td></td>
<td></td>
<td>28.04.2016</td>
<td>1.7.1.0</td>
<td>WAGO / u013972</td>
<td>Replace WagoAppErrorBase with WagoSysErrorBase</td>
</tr>
<tr>
<td>Replace WagoAppErrorBase with WagoSysErrorBase</td>
<td></td>
<td></td>
<td>03.03.2016</td>
<td>1.7.0.0</td>
<td>WAGO / u013972</td>
<td>Add compact function blocks</td>
</tr>
<tr>
<td>Add compact function blocks</td>
<td></td>
<td></td>
<td>02.02.2016</td>
<td>1.6.0.0</td>
<td>WAGO / u013972</td>
<td>Resolve libraries with placeholders</td>
</tr>
<tr>
<td>Resolve libraries with placeholders</td>
<td></td>
<td></td>
<td>29.09.2015</td>
<td>1.5.2.0</td>
<td>WAGO / u013972</td>
<td>Workaround for C0351-Bug</td>
</tr>
<tr>
<td>Workaround for C0351-Bug</td>
<td></td>
<td></td>
<td>23.09.2015</td>
<td>1.5.1.0</td>
<td>WAGO / u013972</td>
<td>Release Version</td>
</tr>
<tr>
<td>Release Version</td>
<td></td>
<td></td>
<td>23.06.2015</td>
<td>1.5.0.0</td>
<td>WAGO / u013972</td>
<td>Release Version</td>
</tr>
</tbody>
</table>
This is a dictionary of all referenced libraries and their name spaces.

**WagoSysAppLED_Internal_PFC**

*Library Identification:*
- Placeholder: WagoSysAppLedInternal
- Default Resolution: WagoSysAppLED_Internal_PFC, * (WAGO)
- Namespace: WagoSysAppLedInternal

*Library Properties:*
- LinkAllContent: False
- Optional: False
- QualifiedOnly: False
- SystemLibrary: False
- PublishSymbolsInContainer: True

**WagoSysBehaviourModels**

*Library Identification:*
- Placeholder: WagoSysBehaviourModels
- Default Resolution: WagoSysBehaviourModels, * (WAGO)
- Namespace: WagoSysBehaviourModels

*Library Properties:*
- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

**WagoSysErrorBase**

*Library Identification:*
- Placeholder: WagoSysErrorBase
- Default Resolution: WagoSysErrorBase, * (WAGO)
- Namespace: WagoSysErrorBase
Library Properties:

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

Library Parameter:
Parameter: RES_LOG_MAX_FILESIZE = 2000
Parameter: RES_LOG_MAX_FILES = 1
Parameter: RES_LOG_MAX_ENTRIES = 200
Parameter: RES_LOG_NAME = ‘WagoAppResultLogger’

WagoSysStandard

Library Identification:
Placeholder: WagoSysStandard
Default Resolution: WagoSysStandard, * (WAGO)
Namespace: WagoSysStandard

Library Properties:

- LinkAllContent: False
- Optional: False
- QualifiedOnly: False
- SystemLibrary: False
- PublishSymbolsInContainer: True

WagoSysVersion

Library Identification:
Name: WagoSysVersion
Version: 1.0.0.0
Company: WAGO
Namespace: WagoSysVersion

Library Properties:

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

WagoTypesAppLED

Library Identification:
Placeholder: WagoTypesAppLED
Default Resolution: WagoTypesAppLED, * (WAGO)
Namespace: WagoTypesAppLED
Library Properties:

- LinkAllContent: False
- Optional: False
- QualifiedOnly: False
- SystemLibrary: False
- PublishSymbolsInContainer: True

WagoTypesCommon

Library Identification:

Placeholder: WagoTypesCommon
Default Resolution: WagoTypesCommon, * (WAGO)
Namespace: WagoTypes

Library Properties:

- LinkAllContent: False
- QualifiedOnly: False
- SystemLibrary: False
- Optional: False

© WAGO Kontakttechnik GmbH & Co. KG, Germany 2018 – All rights reserved. For the avoidance of doubt, this copyright notice does not only apply to the information above but also and primarily to the described library itself. Please note that third-party products are always mentioned without reference to intellectual property rights, including patents, utility models, designs and trademarks, accordingly the existence of such rights cannot be excluded. WAGO is a registered trademark of WAGO Verwaltungsgesellschaft mbH.