CXMM_MessageManagement

Function Block library for PLCnext Engineer

Revision: 1.0.0
Date: 10/06/2021
Publisher: Development ICS
PHOENIX CONTACT GmbH
Flachsmarkstraße 8
D-32825 Blomberg

Table of Contents

1 General Information

Help file for the PLCnext Engineer library CXMM_MessageManagement version 1.0.0.

1.1 Supported PLCnext Engineer versions

Minimal required version: PLCnext Engineer 2020.0 LTS.

1.2 Supported devices

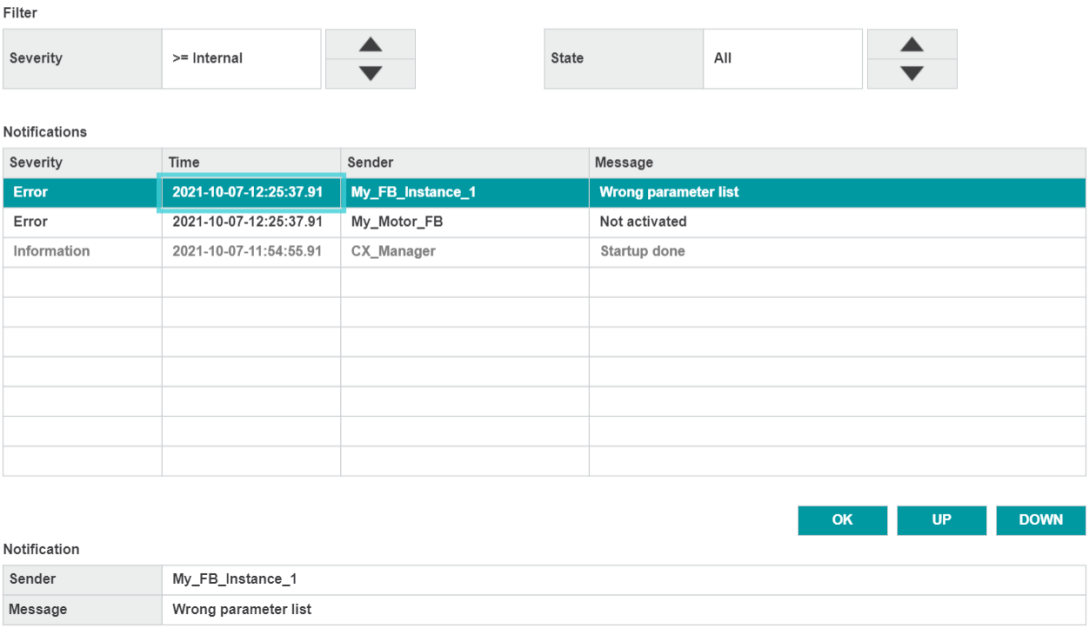
Device	Order Number
AXC F 1152	1151412
AXC F 2152	2404267
AXC F 3152	1069208

1.3 Dependencies

The library depends not on other stuff.

2 Introduction

The CXMM_MessageManagement library offers functions to display information-, warning-, error- and other messages from the application program on the hmi. The design lean on the PLCnext WBM notification display.



CXMM_MessageDisplay

3 First steps example

3.1 Set the message text to the plc variable udtTextBuffer

It is recommended to store the message text in an csv file and to use the FB PBCL_FileCsvRead from the PLCnextBase library to set the data to the pls variable.

3.1.1 Set text for the sender column

For the sender column set the parameters as follow:
uiSender > 0, wCode = 0, wAddCode = 0

```
CXMM_udtMessageMgr.udtTextBuffer.arrText[0].uiSender := 12345;
CXMM_udtMessageMgr.udtTextBuffer.arrText[0].wCode    := WORD#16#0;
CXMM_udtMessageMgr.udtTextBuffer.arrText[0].wAddCode := WORD#16#0;
CXMM_udtMessageMgr.udtTextBuffer.arrText[0].strText  := 'My_FB';
```

3.1.2 Set text for the message column

For the message column set the parameters as follow:
uiSender > 0, wCode > 0, wAddCode > 0

```
CXMM_udtMessageMgr.udtTextBuffer.arrText[1].uiSender := 12345;
CXMM_udtMessageMgr.udtTextBuffer.arrText[1].wCode    := WORD#16#8001;
CXMM_udtMessageMgr.udtTextBuffer.arrText[1].wAddCode := WORD#16#1001;
CXMM_udtMessageMgr.udtTextBuffer.arrText[1].strText  := 'Not in position';
```

3.1.3 Example how to use the PBCL_FileCsvRead FB

```
PBCL_FileCsvRead(
    uiOffset        := UINT#1,
    strFileName     := strFileNameFileCsvRead,
    strDataTypes    := 'UINT;WORD;WORD;STRING' ,
    strDelimiter    := ';',
    udiRowCnt       => udiRowCntFileCsvRead,
    anyTable        := CXMM_udtMessageMgr.udtTextBuffer.arrText);
```

	A	B	C	D
1	uiSender(dez)	wCode(dez)	wAddCode(dez)	strText
2	12345	0	0	My_FB
3	12345	32769	4097	Not in position
4	12345	1	0	Not activated
5	12345	1	2	Wrong parameter list
6	1001	0	0	My_FB_Instance_1
7	1002	0	0	My_FB_Instance_2
8	1003	0	0	My_FB_Instance_3

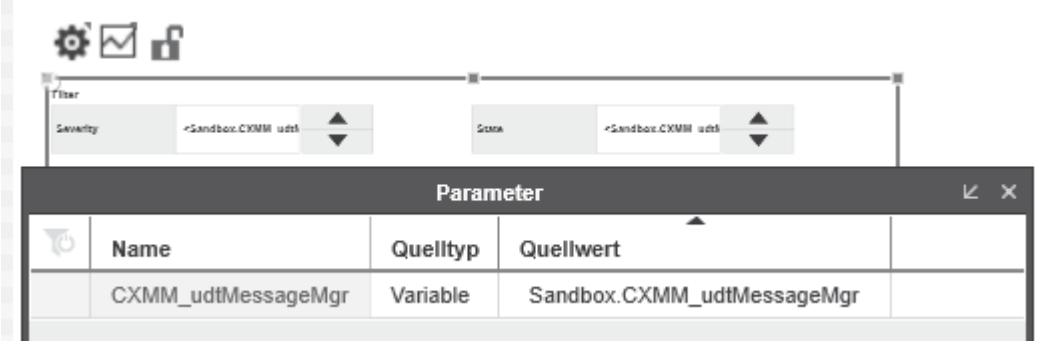
MessageText_csv

3.2 Instantiate the function block CXMM_MessageManager

```
CXMM_MessageManager(udtMessageMgr := CXMM_udtMessageMgr);
```

3.3 Instantiate the hmi symbol CXMM_MessageDisplay

Connect it to the **CXMM_MessageManager** function block.

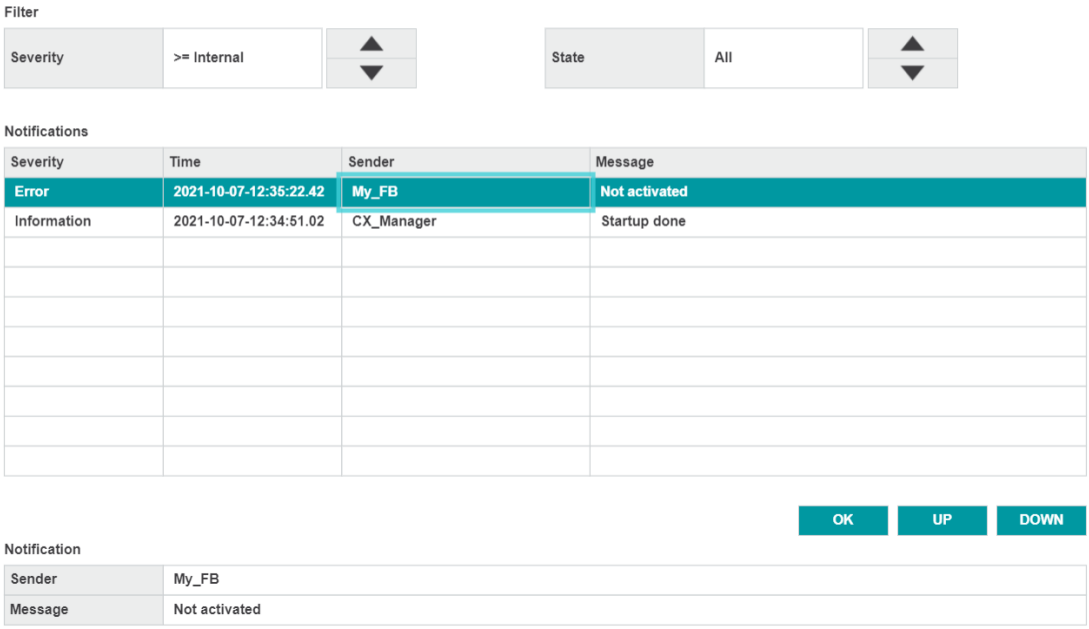


CXMM_MessageDisplayParameter

3.4 Instantiate the function block CXMM_SendMessage

```
CXMM_SendMessage_11(
    xSend           := xSend,
    uiSender        := UINT#12345,
    uiSenderType    := UINT#0,
    wCode           := WORD#1,
    wAddCode        := WORD#0,
    enSeverity      := CXMM_enSeverity#Error,
    udtMessageMgr   := CXMM_udtMessageMgr);
```

3.5 CXMM_MessageDisplay show the message text



CXMM_MessageDisplay

4 Function and Function Blocks

4.1 CXMM_MessageManager

POU type: `Function Block`

Description of the POU.

The function block is the ‘backend’ for the CXMM_MessageDisplay hmi symbol. It forwards the message to it and holds all functions the user has on the CXMM_MessageDisplay hmi symbol.

4.1.1 InOut Parameters

4.1.1.1 udtMessageMgr

Data type: *`CXMM_udtMessageManager`*

Data exchange structure.

4.2 CXMM_SendMessage

POU type: `Function Block`

Description of the POU.

The function block takes an message entry into the buffer.

4.2.1 Input Parameters

4.2.1.1 xSend

Data type: `INT`

Put the message into the buffer on rising edge.

4.2.1.2 uiSender

Data type: `UINT`

Identifier of the instance which sends the message.

4.2.1.3 uiSenderType

Data type: `UINT`

Identifier of the instance type from the sender which sends the message. 0: Not used.

Example

Message text csv:

uiSender	wCode	wAddCode	strText
12345	0	0	My_FB
12345	8001	0	Not in position
12345	1	0	Not activated
1001	0	0	My_FB_Instance_1
1002	0	0	My_FB_Instance_1

The call of the FB CXMM_SendMessage with following parameters

```
CXMM_SendMessage_12(
    xSend           := xSend,
    uiSender        := UINT#1001,
    uiSenderType    := UINT#12345,
    wCode           := WORD#1,
    wAddCode        := WORD#0,
    enSeverity      := CXMM_enSeverity#Error,
    udtMessageMgr   := CXMM_udtMessageMgr);
```

will send this message:

Severity	Time	Sender	Message
Error	2021.10.07-11:56:32.12	My_FB_Instance_1	Not activated

4.2.1.4 wCode

Data type `WORD`

Diag code.

4.2.1.5 wAddCode

Data type: `WORD`

Additional diag code.

4.2.1.6 enSeverity

Data type: `CXMM_enSeverity`

Severity of message. 1: InternalOnly, 2: Information, 3: Warning, 4: Error, 5: Critical Error, 6: Fatal Error.

4.2.1.7 InOut Parameters

4.2.1.8 udtMessageMgr

Data type: *`CXMM_udtMessageManager`*

Data exchange structure.

5 Data types

5.1 CXMM_udtMessageManager

Data type: `STRUCT`

```
CXMM_udtMessageManager : STRUCT
    udtMessageBuffer    : CXMM_udtMessageBuffer;
    udtTextBuffer       : CXMM_udtMessageTextBuffer;
    udtMessageDisplay   : CXMM_udtMessageDisplay;
END_STRUCT;
```

5.1.1 Structure elements

5.1.1.1 udtMessageBuffer

Data type: `STRUCT`

Types for the messaging function.

```
CXMM_enSeverity : (NotDefined, InternalOnly, Information, Warning, Error,
        CriticalError, FatalError) OF INT := Error;
CXMM_enState : (NotDefined, Comming, Acknowledged) OF INT;

CXMM_udtMessage :   STRUCT
    uiSender          : UINT;         // Identifier of the instance which sends the
        message.
    uiSenderType      : UINT;         // Identifier of the instance type from the
        sender which sends the message.
                                      // 0: Not used.
    enSeverity        : CXMM_enSeverity; // Severity of message. 1: Internal, 2:
        Information, 3: Warning, 4: Error,
                                        // 5: Critical Error, 6: Fatal Error
    wCode             : WORD;         // Code of the instance which sends the message.
    wAddCode          : WORD;         // Additional code
    enState           : CXMM_enState;        // Message state 1: comming, 2:
        acknowledged.
    strDateTime       : STRING;       // Date and time of the message
END_STRUCT;

CXMM_arrMessage : ARRAY[0..99] OF CXMM_udtMessage;

// Structure to hold the current messages. It's a ringbuffer.
CXMM_udtMessageBuffer : STRUCT
    arrMessage        : CXMM_arrMessage;   // message array *)
    uiArrMessageMax   : UINT := 99;      // upper array bound of the message array. *)
    uiPointer         : UINT;            // pointer of last message entry in the
        message array. *)
END_STRUCT;
```

5.1.1.2 udtTextBuffer

Data type: `STRUCT`

Types to handle the message texts.

```
CXMM_udtMessageText : STRUCT
    uiSender    : UINT;
    wCode       : WORD;
    wAddCode    : WORD;
    strText     : STRING;   // the message text
END_STRUCT;

CXMM_arrMessageText   : ARRAY[0..99] OF CXMM_udtMessageText;

CXMM_udtMessageTextBuffer : STRUCT
    arrText             : CXMM_arrMessageText;  // Hold the text of the message
    uiArrTextMax        : INT := 99;            // Upper bound of CXMM_arrMessageText
END_STRUCT;
```

5.1.1.3 udtMessageDisplay

Data type: `STRUCT`

Type for the message line in the message display.

```
CXMM_udtMessageLine : STRUCT
    strSenderText       : STRING;       // Text of sender.
    strMessageText      : STRING;       // Text of message.
    strDataTime         : STRING;       // Time of message is occured.
    strSeverity         : STRING;       // Severity of message -> 1: Internal, 2:
        Information, 3: Warning, 4: Error,
                                        // 5: Critical Error, 6: Fatal Error
    iState              : INT;          // Message state -> 1: comming(new message), 2:
        acknowledged(button OK was pressed)
    xMarked             : BOOL;         // Line is marked
    usiFontColor        : USINT;        // Font color of line -> 0: default, 1: marked,
        2: acknowledged
END_STRUCT;

CXMM_arrMessageLine : ARRAY[0..9] OF CXMM_udtMessageLine;

// Type for the large line. It shows the full text of the marked message.
CXMM_udtLargeLine : STRUCT
    strSenderText   : STRING;
    strMessageText  : STRING;
END_STRUCT;

// Type for the message display
CXMM_udtMessageDisplay : STRUCT
    arrLine             : CXMM_arrMessageLine;
    uiArrLineMax        : UINT := 9;        // Max index for arrLine.
    iShowSeverity       : INT := 1;         // Show only messages which has the
        severity >= the filter value for severity
    iShowState          : INT := 0;         // Show messages with state 0: show all, 1:
        comming(new message),
                                            // 2: acknowledged(button OK was pressed)
    xAcknowledge        : BOOL;            // Acknowledge all messages
    xShowMsgBefore      : BOOL;             // Show messages before
    xShowMsgAfter       : BOOL;             // Show messages after
    udtLargeLine        : CXMM_udtLargeLine;  // Large line for the marked message
    strShowSeverity     : STRING;           // Displayed name for message severity at
        the filter parameter
    strShowState        : STRING;           // Displayed name for message state at the
        filter parameter
END_STRUCT;
```

6 Support

Owner:

Development ICS
Industrial Components and Electronics
Flachsmarktstraße 8
D-32825 Blomberg