# XpsuSupport

## Library Guide

## for XPSU Safety Modules

EIO0000004435.00
04/2021

www.se.com

# Legal Information

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this guide are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owners.

This guide and its content are protected under applicable copyright laws and furnished for informational use only. No part of this guide may be reproduced or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), for any purpose, without the prior written permission of Schneider Electric.

Schneider Electric does not grant any right or license for commercial use of the guide or its content, except for a non-exclusive and personal license to consult it on an "as is" basis. Schneider Electric products and equipment should be installed, operated, serviced, and maintained only by qualified personnel.

As standards, specifications, and designs change from time to time, information contained in this guide may be subject to change without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this material or consequences arising out of or resulting from the use of the information contained herein.
© 2021 Schneider Electric. All rights reserved.

# Table of Contents

# Safety Information

## Important Information

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

| DANGER |
|---|
| **DANGER** indicates a hazardous situation which, if not avoided, **will result in** death or serious injury. |

| WARNING |
|---|
| **WARNING** indicates a hazardous situation which, if not avoided, **could result in** death or serious injury. |

| CAUTION |
|---|
| **CAUTION** indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury. |

| *NOTICE* |
|---|
| ***NOTICE*** is used to address practices not related to physical injury. |

## Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

## Before You Begin

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

**⚠WARNING**

**UNGUARDED EQUIPMENT**

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

**NOTE:** Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

## Start-up and Test

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check are made and that enough time is allowed to perform complete and satisfactory testing.

**⚠WARNING**

**EQUIPMENT OPERATION HAZARD**

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

**Software testing must be done in both simulated and real environments.**

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

## Operation and Adjustments

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer’s instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer’s instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

# About the Book

## Document Scope

The present document describes the function blocks *FB_XpsuDiag* and *FB_XpsuMain* provided by the XpsuSupport library. The library is used in conjunction with safety modules of the XPSU range. It provides diagnostics functionality and performs calculations concerning the service life and due dates for specific types of maintenance of XPSU safety modules as well as connected equipment.

## Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V2.0.

The characteristics that are described in the present document, as well as those described in the documents included in the Related Documents section below, can be found online. To access the information online, go to the Schneider Electric home page www.se.com/ww/en/download/.

The characteristics that are described in the present document should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the document and online information, use the online information as your reference.

## Related Documents

| Document title | Reference |
|---|---|
| EcoStruxure Machine Expert Functions and Libraries User Guide | EIO0000002829 (eng)<br>EIO0000002830 (fre)<br>EIO0000002831 (ger)<br>EIO0000002832 (ita)<br>EIO0000002833 (spa)<br>EIO0000002834 (chi) |
| EcoStruxure Machine Expert Programming Guide | EIO0000002854 (eng)<br>EIO0000002855 (fre)<br>EIO0000002856 (ger)<br>EIO0000002857 (ita)<br>EIO0000002858 (spa)<br>EIO0000002859 (chi) |
| XPSUABx1C User Guide | EIO0000003454 (eng)<br>EIO0000003455 (fre)<br>EIO0000003456 (ger)<br>EIO0000003457 (ita)<br>EIO0000003458 (spa)<br>EIO0000003461 (chi) |
| XPSUAFx3A User Guide | EIO0000003465 (eng)<br>EIO0000003466 (fre)<br>EIO0000003467 (ger) |

| Document title | Reference |
|---|---|
| | EIO0000003468 (ita)<br>EIO0000003469 (spa)<br>EIO0000003472 (chi) |
| XPSUAKx2A User Guide | EIO0000003476 (eng)<br>EIO0000003477 (fre)<br>EIO0000003478 (ger)<br>EIO0000003479 (ita)<br>EIO0000003480 (spa)<br>EIO0000003483 (chi) |
| XPSUATx3A3A User Guide | EIO0000003443 (eng)<br>EIO0000003444 (fre)<br>EIO0000003445 (ger)<br>EIO0000003446 (ita)<br>EIO0000003447 (spa)<br>EIO0000003450 (chi) |
| XPSUDNx3A User Guide | EIO0000003498 (eng)<br>EIO0000003499 (fre)<br>EIO0000003500 (ger)<br>EIO0000003501 (ita)<br>EIO0000003502 (spa)<br>EIO0000003505 (chi) |
| XPSUSx2A User Guide | EIO0000003487 (eng)<br>EIO0000003488 (fre)<br>EIO0000003489 (ger)<br>EIO0000003490 (ita)<br>EIO0000003491 (spa)<br>EIO0000003494 (chi) |
| XPSUVNx1A User Guide | EIO0000004260 (eng)<br>EIO0000004262 (fre)<br>EIO0000004261 (ger)<br>EIO0000004264 (ita)<br>EIO0000004263 (spa)<br>EIO0000004265 (chi) |
| XPSUABx1C Instruction Sheet | PHA71839 (eng, fre, ger, ita, spa, chi) |
| XPSUABx1C Instruction Sheet | PHA71840 (eng, jpn, kor, por, rus, tur) |
| XPSUAFx3A Instruction Sheet | PHA71842 (eng, fre, ger, ita, spa, chi) |
| XPSUAFx3A Instruction Sheet | PHA71843 (eng, jpn, kor, por, rus, tur) |
| XPSUAKx2A Instruction Sheet | PHA71845 (eng, fre, ger, ita, spa, chi) |
| XPSUAKx2A Instruction Sheet | PHA71846 (eng, jpn, kor, por, rus, tur) |
| XPSUATx3A3A Instruction Sheet | PHA71829 (eng, fre, ger, ita, spa, chi) |
| XPSUATx3A3A Instruction Sheet | PHA71837 (eng, jpn, kor, por, rus, tur) |
| XPSUDNx3A Instruction Sheet | PHA71850 (eng, fre, ger, ita, spa, chi) |
| XPSUDNx3A Instruction Sheet | PHA71851 (eng, jpn, kor, por, rus, tur) |

| Document title | Reference |
|---|---|
| XPSUSx2A Instruction Sheet | PHA71847 (eng, fre, ger, ita, spa, chi) |
| XPSUSx2A Instruction Sheet | PHA71849 (eng, jpn, kor, por, rus, tur) |
| XPSUVNx1A Instruction Sheet | NNZ32597 (eng, fre, ger, ita, spa, chi) |
| XPSUVNx1A Instruction Sheet | NNZ32602 (eng, jpn, kor, por, rus, tur) |

## Product Related Information

**⚠WARNING**

**LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

Before you attempt to provide a solution (machine or process) for a specific application using the POUs found in the library, you must consider, conduct and complete best practices. These practices include, but are not limited to, risk analysis, functional safety, component compatibility, testing and system validation as they relate to this library.

**⚠WARNING**

**IMPROPER USE OF PROGRAM ORGANIZATION UNITS**

- Perform a safety-related analysis for the application and the devices installed.
- Ensure that the Program Organization Units (POUs) are compatible with the devices in the system and have no unintended effects on the proper functioning of the system.
- Use appropriate parameters, especially limit values, and observe machine wear and stop behavior.
- Verify that the sensors and actuators are compatible with the selected POUs.
- Thoroughly test all functions during verification and commissioning in all operation modes.
- Provide independent methods for critical control functions (emergency stop, conditions for limit values being exceeded, etc.) according to a safety-related analysis, respective rules, and regulations.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠WARNING**

**UNINTENDED EQUIPMENT OPERATION**

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Care must be taken and provisions made for use of this library for machine control to avoid inadvertent consequences of commanded machine operation, state changes, or alteration of data memory or machine operating elements.

**⚠WARNING**

**UNINTENDED EQUIPMENT OPERATION**

- Place operator devices of the control system near the machine or in a place where you have full view of the machine.
- Protect operator commands against unauthorized access.
- If remote control is a necessary design aspect of the application, ensure that there is a local, competent, and qualified observer present when operating from a remote location.
- Configure and install the Run/Stop input, if so equipped, or, other external means within the application, so that local control over the starting or stopping of the device can be maintained regardless of the remote commands sent to it.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Update your application program as required, paying particular attention to I/O address adjustments, whenever you modify the hardware configuration.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**

## Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

| Standard | Description |
|---|---|
| IEC 61131-2:2007 | Programmable controllers, part 2: Equipment requirements and tests. |
| ISO 13849-1:2015 | Safety of machinery: Safety related parts of control systems.<br><br>General principles for design. |
| EN 61496-1:2013 | Safety of machinery: Electro-sensitive protective equipment.<br><br>Part 1: General requirements and tests. |
| ISO 12100:2010 | Safety of machinery - General principles for design - Risk assessment and risk reduction |
| EN 60204-1:2006 | Safety of machinery - Electrical equipment of machines - Part 1: General requirements |
| ISO 14119:2013 | Safety of machinery - Interlocking devices associated with guards - Principles for design and selection |
| ISO 13850:2015 | Safety of machinery - Emergency stop - Principles for design |
| IEC 62061:2015 | Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems |
| IEC 61508-1:2010 | Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements. |
| IEC 61508-2:2010 | Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems. |
| IEC 61508-3:2010 | Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements. |
| IEC 61784-3:2016 | Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions. |
| 2006/42/EC | Machinery Directive |
| 2014/30/EU | Electromagnetic Compatibility Directive |
| 2014/35/EU | Low Voltage Directive |

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

| Standard | Description |
|---|---|
| IEC 60034 series | Rotating electrical machines |
| IEC 61800 series | Adjustable speed electrical power drive systems |
| IEC 61158 series | Digital data communications for measurement and control – Fieldbus for use in industrial control systems |

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

**NOTE:** The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

# Presentation of the Library

## General Information

### Library Overview

The XpsuSupport library provides the function blocks *FB_XpsuDiag* and *FB_XpsuMain* for diagnostics and maintenance purposes with XPSU safety modules.

XPSU safety modules generate bit sequences that are available at the non-safety-related output Z1. The bit sequences encode diagnostics information relating to the XPSU safety module. Refer to the user guide of your XPSU safety module for details on the output Z1, Related Documents, page 8.

If the output Z1 is connected to a controller or to other suitable equipment, the bit sequence can be used as input information for the function block *FB_XpsuDiag*. The function block *FB_XpsuDiag* decodes the bit sequence provided by the XPSU safety module and outputs a diagnostics code.

This diagnostics code can be made available to the function block *FB_XpsuMain*.

The function block *FB_XpsuMain* provides counters that let you track and manage maintenance and service life requirements for the XPSU safety module itself as well as connected equipment such as sensors, command devices, and actuators:

- Counters for cycles of the safety-related inputs
- Counters for cycles of the safety-related outputs
- Counter for cycles of the safety module itself

In addition, the function block can be used to monitor and manage the due dates of proof tests required for the safety-related function and/or the equipment used to implement it.

### General Considerations

The bit sequence generated by the XPSU safety module consists of 10 bits. The bit duration set at the input *i_timBitDuration* of the function block *FB_XpsuDiag* must be supported by the bit duration of the connected XPSU safety module. Refer to the user guide of your XPSU safety module for details, Related Documents, page 8. For correct detection of the full bit sequence, the cycle time of the task that executes the function blocks must be less than or equal to the signal duration of one bit divided by four. For example, if the bit duration is set to 200 ms, the cycle time must be less than or equal to 50 ms.

Both function blocks of the library must be used in a cyclic task and executed in each cycle of their task.

The duration of the full bit sequence is ten times the adjusted bit duration. Changes in state (for example, deactivation of a safety-related input of the XPSU safety module) during a running sequence are transmitted in the subsequent sequence. If such changes in state do not persist until the next sequence starts (for example, if a safety-related input is deactivated and activated within the duration of a single bit sequence), they are not encoded in this bit sequence. In such a case, they cannot be detected by the function block *FB_XpsuDiag*. As a consequence, the corresponding cycle counter is not increased by the function block *FB_XpsuMain*.

If several devices are connected to the safety-related inputs of an XPSUDN or an XPSUS safety module, the cycle counter may miss cycles under the following condition: If one connected device has deactivated a safety-related input and not yet reactivated it, and if other devices connected to other safety-related inputs perform switching operations in the meantime.

Removing power to the XPSU safety module is detected by the library with a delay of up to eight seconds.

Removing power to the XPSU safety module may cause the function block *FB_XpsuDiag* to detect information that is incorrect on the state of the XPSU safety module.

## Characteristics of the Library

| Characteristic | Value |
|---|---|
| Library title | XpsuSupport |
| Company | Schneider Electric |
| Category | Application |
| Component | CoreLibraries |
| Default namespace | XPSU |
| Language model attribute | qualified-access-only |
| Forward compatible library | Yes (FCL) |

**NOTE:** For this library, qualified-access-only is set. This means that the POUs, data structures, enumerations, and constants have to be accessed using the namespace of the library.

# Enumerations

## *ET_ModuleType* - General Information

### Overview

| | |
|---|---|
| Type: | List type |
| Available as of: | V1.0.4.0 |

### Description

This enumeration provides a list of the XPSU safety modules that can be selected for the input *i_etModuleType* of the function block *FB_XpsuDiag*.

### Enumeration Elements

| **Name** | **Value (UDINT)** | **Description** |
|---|---|---|
| — | 0 | No XPSU safety module selected. |
| *XPSUAB* | 1 | Safety module XPSUAB |
| *XPSUAF* | 2 | Safety module XPSUAF |
| *XPSUAK* | 3 | Safety module XPSUAK |
| *XPSUAT* | 4 | Safety module XPSUAT |
| *XPSUDN* | 5 | Safety module XPSUDN |
| *XPSUS* | 6 | Safety module XPSUS |
| *XPSUVN* | 7 | Safety module XPSUVN |

If the value is 0 or if no value is set for the input *i_etModuleType*, the function block *FB_XpsuDiag* detects an error.

# Function Blocks

## *FB_XpsuDiag* - General Information

### Overview

| | |
|---|---|
| Type: | Function block |
| Available as of: | V1.0.4.0 |
| Inherits from: | - |
| Implements: | - |

### Functional Description

The function block *FB_XpsuDiag* decodes the bit sequence provided at the input *i_xDiagSignal*. This bit sequence is provided by an XPSU safety module via its non-safety-related output Z1. The bit sequence encodes diagnostics information on the safety module (see *Output q_dwStatus*, page 19 for a list of codes).

The function block needs to be activated by setting the value at its input *i_xEnable* to TRUE. If the value at the input *i_xEnable* is set to FALSE, the bit sequence is not analyzed and the outputs of the function block are set to their default values.

The XPSU safety module used is specified via the input *i_etModuleType*.

The bit duration is set via the input *i_timBitDuration*.

The cycle time of the task in which the function block is called has to be specified at the input *i_timTaskCycle*. See chapter *General Considerations*, page 14 for details.

The function block must be used in a cyclic task, and it has to be executed in each cycle of this task. If the function block is not called in a cycle of the task, an error is detected (error code `16#1000`)

The diagnostics data decoded from the bit sequence is provided at the output *q_stDiagCode* (structure *ST_DiagCodes*) of the function block. Depending on the analysis result, the corresponding structure element in the structure *ST_DiagCodes* is set to TRUE.

The output *q_dwStatus* provides the 6-bit code part (without the 4-bit start part) of the last bit sequence as a double word.

If the value at the input *i_xDiagSignal* is static TRUE, the output *q_xComWireInShort* is set to TRUE. This indicates a possible cross circuit in the wiring between the output Z1 of the XPSU safety module and the connected input of the controller. If the value at the input *i_xDiagSignal* is static FALSE, the value of the output *q_xComWireOpen* is set to TRUE.

The outputs *q_xBusy* and *q_xValid* provide information on the status of the function block. The outputs *q_xError* and *q_wErrorId* provide information on detected errors.

### Interface

| Input | Data type | Description |
|---|---|---|
| *i_xEnable* | BOOL | TRUE activates the function block. When it is activated, the bit sequence at input *i_xDiagSignal* is evaluated. |
| *i_xDiagSignal* | BOOL | Input for connecting the bit sequence provided by the XPSU safety module.<br><br>Connect this input to the variable which is assigned to the signal from the non- |

| Input | Data type | Description |
|---|---|---|
| | | safety-related output Z1 of the XPSU safety module. |
| *i_timTaskCycle* | TIME | Cycle time of the cyclic task in which the function block is executed.<br>**NOTE:** The cycle time must be less than or equal to the signal duration of one bit (set via *i_timBitDuration*) divided by four. Example: If the bit duration is set to 200 ms, the cycle time must be less than or equal to 50 ms. See chapter *General Considerations*, page 14 for details. |
| *i_etModuleType* | ET_ModuleType, page 16 | Sets the type of XPSU safety module from which the bit sequence is received. If the input is not connected, or if the value is invalid, the error code 16x1007 is provided at the output *q_wErrorId*.<br>Default value: 0 |
| *i_timBitDuration* | TIME | Sets the duration of a bit in milliseconds. The duration of the full sequence of ten bits is ten times this value.<br>Value range: 50 ... 200<br>Default value: 200 |

| Output | Data type | Description |
|---|---|---|
| *q_xBusy* | BOOL | If the value at this output is TRUE, the function block is being executed. |
| *q_xError* | BOOL | If the value at this output is TRUE, the function block has detected an error. Refer to *q_wErrorId* for details. |
| *q_wErrorId* | WORD | Provides information on detected errors. Error codes are listed in the section *Error Codes*, page 21. |
| *q_xValid* | BOOL | If the value at this output is TRUE, the values of the output variables are valid. |
| *q_stDiagCode* | ST_DiagCodes | Diagnostics information decoded from the input bit sequence. The corresponding bit value in the structure *ST_DiagCodes* is set to TRUE. |
| *q_dwStatus* | DWORD | Provides the 6-bit code part (without the 4-bit start part) of the last detected bit sequence converted to a double word data type.<br>Refer to Output q_dwStatus, page 19 for details. |

| Output | Data type | Description |
|---|---|---|
| *q_xComWireInShort* | BOOL | If the value at the input *i_xDiagSignal* is static TRUE, the value at this output is set to TRUE. This indicates a possible cross circuit in the wiring between the output Z1 of the XPSU safety module and the connected input of the controller.<br><br>Verify correct wiring between the output Z1 of the XPSU safety module and the controller as well as correct operation of the output Z1 and the input of the controller. |
| *q_xComWireOpen* | BOOL | If the value at the input *i_xDiagSignal* is static FALSE, the value at this output is set to TRUE. This indicates that either the output Z1 of the XPSU safety module is not correctly connected to the controller input or that the XPSU safety module is not energized.<br><br>Verify correct wiring between the output Z1 of the XPSU safety module and the controller as well as correct operation of the output Z1 and the input of the controller. |

## Output q_dwStatus

The bit sequence consists of 10 bits. The first 4 bits (`0010`) represent the start sequence, followed by a 6-bit code sequence.

The following table lists the 6-bit code part of the diagnostics codes available at the output *q_dwStatus*.

| Code part of bit sequence | Decimal value at *q_dwStatus* | Description | Class. (1) |
|---|---|---|---|
| `101111` | 47 | Device in operating state Run, safety-related outputs activated. | S |
| `101110` | 46 | Start input activated. Waiting for falling edge for monitored start. | S |
| `101010` | 42 | Waiting for rising edge for automatic/manual or monitored start. | S |
| `101011` | 43 | Waiting for start-up test. | S |
| `101001` | 41 | Input S63 is expected to change its state. | S |
| `101000` | 40 | Input S62 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S62 and S63 are expected to change their state. | S |
| `111000` | 56 | Input S53 is expected to change its state. | S |
| `111001` | 57 | Input S52 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S52 and S53 are expected to change their state. | S |
| `111011` | 59 | Input S43 is expected to change its state. | S |
| `111010` | 58 | Input S42 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S42 and S43 are expected to change their state. | S |
| `111110` | 62 | Input S33 is expected to change its state. | S |
| `111111` | 63 | Input S32 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S32 and S33 are expected to change their state.<br><br>XPSUVN: Voltage U32 does not meet the requirements for detected standstill while U12 already does. | S |
| `111101` | 61 | Input S23 is expected to change its state. | S |
| `111100` | 60 | Input S22 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S22 and S23 are expected to change their state. | S |

| Code part of bit sequence | Decimal value at *q_dwStatus* | Description | Class. (1) |
|---|---|---|---|
| 110100 | 52 | Input S13 is expected to change its state. | S |
| 110101 | 53 | Input S12 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S12 and S13 are expected to change their state.<br><br>XPSUVN: Voltage U12 does not meet the requirements for detected standstill while U32 already does. | S |
| 110111 | 55 | Safety-related inputs deactivated, safety-related outputs deactivated.<br><br>XPSUVN: Voltage at safety-related input is above the adjusted voltage threshold, device is in defined safe state. | S |
| 110110 | 54 | Instantaneous safety-related outputs are deactivated, delayed safety-related outputs are still activated. | S |
| 100111 | 39 | Synchronization alert. Both synchronized safety-related inputs have been activated, but not within the synchronization time. | E |
| 110011 | 51 | Synchronization alert. One of the synchronized safety-related inputs is still deactivated, but the synchronization time has already elapsed. | E |
| 100000 | 32 | Antivalence alert at input S2x.<br><br>XPSUVN: Wiring in circuit for voltage U32 interrupted (between L3 and L2, wire break). | E |
| 100110 | 38 | Antivalence alert at input S1x.<br><br>XPSUVN: Wiring in circuit for voltage U12 interrupted (between L1 and L2, wire break). | E |
| 100011 | 35 | Cross-circuit detected at input used for Cancel Delay function. | E |
| 110000 | 48 | Cross-circuit detected at start input. | E |
| 011100 | 28 | Cross-circuit detected at input S63. | E |
| 011101 | 29 | Cross-circuit detected at input S62. | E |
| 011111 | 31 | Cross-circuit detected at input S53. | E |
| 011110 | 30 | Cross-circuit detected at input S52. | E |
| 011010 | 26 | Cross-circuit detected at input S43. | E |
| 011011 | 27 | Cross-circuit detected at input S42. | E |
| 101100 | 44 | Cross-circuit detected at input S33. | E |
| 011000 | 24 | Cross-circuit detected at input S32. | E |
| 001110 | 14 | Cross-circuit detected at input S23. | E |
| 001111 | 15 | Cross-circuit detected at input S22. | E |
| 001101 | 13 | Cross-circuit detected at input S13. | E |
| 001100 | 12 | Cross-circuit detected at input S12. | E |
| 000111 | 7 | Configuration error detected. | E |
| 000110 | 6 | General error detected in expansion module. | E |
| 000011 | 3 | General error detected. | E |
| 101101 | 45 | Supply voltage is out of tolerance. If the supply voltage drops below the lower limit value, the output Z1 is no longer supplied. | E |

(1) Classification of the message: E = Error, S = Status

The meaning of the bit sequences are specific to the XPSU safety module used in your application. Refer to the XPSU safety modules user guides, page 8 for details. For example, 100110 can indicate an antivalence error (xErrorS1x) for an XPSUDN safety module or a wire break condition (xOpenWireL1L2) for an XPSUVN safety module.

## Error Codes

If the function block detects an error, the output *q_xError* is set to TRUE and an error code is provided at the output *q_wErrorId*.

| Error code | Description |
|---|---|
| `16#1000` | Function block is not called cyclically.<br><br>**Remedy**: Call the function block in a cyclic task and verify that the function block call cannot be skipped (for example, by a preceding conditional jump). |
| `16#1001` | Invalid task cycle time at the input *i_timTaskCycle*.<br><br>**Remedy**: Set the cycle time to a value less than or equal to the duration of one bit (set via *i_timBitDuration*) divided by four. Refer to chapter General Considerations, page 14 for details. |
| `16#1002` | Synchronization with the bit sequence provided by the XPSU safety module not successful.<br><br>**Remedy**: Verify that a valid signal from the output Z1 of the XPSU safety module) is properly connected to the input of the function block and that the XPSU safety module is operational. Verify that the function block is properly configured for the XPSU safety module used. |
| `16#1004` | The maximum number of invalid bit sequences has been exceeded.<br><br>**Remedy**: Verify that a valid signal from the output Z1 of the XPSU safety module) is properly connected to the input of the function block and that the XPSU safety module is operational. |
| `16#1005` | Internal function block error detected.<br><br>**Remedy**: Disable and re-enable the function block. |
| `16#1006` | Incorrect bit duration.<br><br>**Remedy**: Set the bit duration at the input *i_timBitDuration* to a value between 50 and 200 ms. |
| `16#1007` | Invalid XPSU safety module.<br><br>**Remedy**: Set the value for the XPSU safety module at the input *i_etModuleType* to a value corresponding to the XPSU safety module to be connected in the enumeration *ET_ModuleType*, page 16. |

# *FB_XpsuMain* - General Information

## Overview

| | |
|---|---|
| Type: | Function block |
| Available as of: | V1.0.4.0 |
| Inherits from: | - |
| Implements: | - |

## Functional Description

The function block *FB_XpsuMain* evaluates the diagnostics codes decoded by the function block *FB_XpsuDiag*. Based on the diagnostics codes, the function block performs calculations that let you track and manage maintenance and service life requirements (for example, due dates for proof tests or replacements) for the XPSU safety module itself as well as connected equipment such as sensors, command devices, and actuators. The function block provides counters for the number of remaining cycles:

- Counters for remaining cycles of the safety-related input channels: One cycle corresponds to one activation of a safety-related input channel.
- Counters for remaining cycles of the safety-related outputs: One cycle corresponds to one activation of a safety-related output.
- Counter for remaining cycles of the safety module itself: Once cycle corresponds to one transition from the defined safe state to the defined non-safe state of the XPSU safety module.

**NOTE:** Refer to the user guide of your XPSU safety module, page 8 for details on the use of the terms activation and deactivation, and on the defined safe state.

The counters allow you to monitor the number of operations of equipment connected to the XPSU safety module as a function of the cycles of the safety-related inputs and the safety-related outputs.

**NOTE:** The present documents refers to the number of “cycles of the safety-related input channels” and the number of “cycles of the safety-related outputs” as defined above. Whether or not this corresponds to the number of cycles of the equipment connected depends on the definition of a cycle for such equipment. For example, if a device connected to a safety-related input counts one “off-on” and one “on-off” operation as two separate cycles, this corresponds to one cycle of the safety-related input (one activation) of the XPSU safety module.

Typical scenarios for managing the counters include replacements of input devices or output devices as well as a replacement of the XPSU safety module itself. If you replace the XPSU safety module and activate a new XPSU safety module via the input *i_xNewModuleActive*, the counter for the total number of cycles is reset to zero. Since a replacement of the XPSU safety module may not necessitate a replacement of the connected equipment, the values of the counters for the safety-related inputs and the safety-related outputs retain the values at the time of the replacement. If you need to replace connected equipment or add further equipment, you can reset the corresponding counters to zero.

In addition, the function block can be used to monitor and manage the due dates of proof tests required for the safety-related function and/or the equipment used to implement it.

Both function blocks *FB_XpsuMain* and *FB_XpsuDiag* must be called in the same task cycle.

In order to evaluate the diagnostics code received from the function block *FB_XpsuDiag*, the function block *FB_XpsuMain* needs to be activated by setting the value at its input *i_xEnable* to TRUE. If the value at the input *i_xEnable* is set to FALSE, the diagnostics codes are no longer evaluated and the outputs of the function block are set to their default values.

The input *i_dwStatus* is connected to the output *q_dwStatus* of the function block *FB_XpsuDiag*. The input *i_xValid* is connected to the output *q_xValid* of the function block *FB_XpsuDiag*.

The function block receives the diagnostics information serving as the basis for the calculations via its input *i_dwStatus*.

## Configuration of the Function Block

To be able to calculate the remaining number of cycles of the safety-related inputs, the safety-related outputs and the safety module itself, the function block requires the following information:

- The safety-related inputs and the safety-related outputs to be monitored.
- The maximum number of cycles for each safety-related input, each safety-related output and the XPSU safety module itself. These values are taken as the start values and decreased by each cycle detected by the function block *FB_XpsuDiag*.

  **NOTE:** Cycles may not be detected if activation and deactivation occur too fast or if several inputs switch at the same time. Refer to *General Considerations*, page 14 for details.

The required information is provided in the form of data structures connected to the corresponding inputs of the function block. These data structures map the hardware implementation, that is, the input devices/sensors connected to the safety-related inputs and the actuators connected to the safety-related outputs of the XPSU safety module:

- The structure *ST_InputControl* is provided for the counters for the cycles of the safety-related inputs. The structure configures the input channels (CH+ (or CH- in the case of XPSUAK and XPSUAT)) of the safety-related inputs of the XPSU safety module to which the input devices/sensors are connected. An array of this structure with the length of six has to be connected to the input *i_astControlInp*.
- The structure *ST_DevControl* is provided for the counters for the safety-related outputs and for the counter for the XPSU safety module itself.
  - The structure has to be connected to the input *i_stControlProc* (counter for the cycles of the XPSU safety module itself).
  - An array of the structure with the length of 13 has to be connected to the input *i_astControlOut* (counters for the cycles of the safety-related outputs).

The structure *ST_RemainNumOp* provides the information on the remaining number of cycles for the safety-related inputs, the safety-related outputs, and the XPSU safety module itself. This structure has to be connected to the output *q_stRemainNumOp* of the function block.

**NOTE:** To keep the counters from being reset, declare the related variables as persistent variables. See *Persistent Variables*, page 27 for details.

The output *q_xOpExceeded* is set toTRUE if at least one structure element of the structure *ST_RemainNumOp* has the value 0. This means that no cycles are left for the corresponding safety-related input, safety-related output, or the XPSU safety module itself. The value at the output *q_wExceededId* indicates the affected safety-related input or safety-related output, or the XPSU safety module itself.

The output *q_udiNumOpSystem* provides the total number of cycles of the XPSU safety module.

## Proof Tests

Safety-related functions and the hardware used for their implementation require proof tests at specific intervals. The function block lets you track and manage such proof test. The date and time of the proof test are set via the input *i_dtTestIntervalRef*. The interval between consecutive proof tests is set via the input *i_udiTestInterv*. After having performed a proof test, confirm this with a rising edge at the input *i_xTestExecuted*. The confirmed date for the first proof test is available at the output *q_datTestStart*. The due date for the next proof test is provided at the output *q_datNextProof*. The function block provides time stamps of the last ten proof tests in an array at the output *q_adtProofTests*.

If you add or replace an XPSU safety module, confirm this with a rising edge at the input *i_xNewModuleActive*. This confirmation also resets the time stamp array, enters the time stamp in the first array element, and updates the value at the output *q_datTestStart*.

The value TRUE at the output *q_xTestIntervViol* indicates that the specified interval has been exceeded, that is, the proof test has not been performed on schedule.

Proof test time stamps can also be loaded from an external proof test array, for example, if the XPSU safety module is connected to a different controller. Such arrays can, for example, originate from an HMI application. To load an external time stamp array, set the input *i_xSaveBProofTest* to TRUE. As a result, the array connected to the input *i_adtProofTests* is read and copied to the internal array.

If the value at the output *q_xActive* is TRUE, the function block is activated. The outputs *q_xError* and *q_wErrorId* provide information on detected errors.

## Interface

| **Input** | **Data type** | **Description** |
|---|---|---|
| *i_xEnable* | BOOL | TRUE activates the function block. |
| *i_dwStatus* | DWORD | Connect this input to the output *q_dwStatus* of the function block *FB_XpsuDiag*.<br><br>The function block processes this status information and writes the data to the structures connected to the outputs. |
| *i_xValid* | BOOL | Connect this input to the output *q_xValid* of the function block *FB_XpsuDiag*. |
| *i_stControlProc* | ST_DevControl | The structure *ST_DevControl* connected to this input sets the maximum number of cycles for the XPSU safety module itself. The structure element *xReset* can be used to reset the corresponding counter. |
| *i_astControlInp* | ARRAY[1..6] OF ST_InputControl | The array of the structure *ST_InputControl* connected to this input configures the safety-related inputs of the XPSU safety module. It identifies the connected input channels of the safety-related inputs and sets the maximum number of cycles for the safety-related inputs. The structure element *xReset* can be used to reset the corresponding counters. |
| *i_astControlOut* | ARRAY[1..13] OF ST_DevControl | The array of the structure *ST_DevControl* connected to this input configures the safety-related outputs of the XPSU safety module. It identifies the connected safety-related outputs and sets the maximum number of cycles for the corresponding relay contacts. The structure element *xReset* can be used to reset the corresponding counters.<br><br>The number of 13 array elements corresponds to the maximum of the seven safety-related outputs that an XPSU safety module can have plus six safety-related outputs provided by an XPSUEP extension module. |
| *i_dtTestIntervalRef* | DATE_AND_TIME | Date and time of the proof test to be performed (reference to the controller time). |

| Input | Data type | Description |
|---|---|---|
| *i_udiTestInterv* | UDINT | Proof test interval. Specify the number of days between consecutive proof tests. |
| *i_xNewModuleActive* | BOOL | A rising edge at this input confirms that an XPSU safety module has been added or replaced.<br><br>Results<br>• The array with the proof test time stamps connected to the output *q_adtProofTests* is reset.<br>• A time stamp is written to the first array element of the array with the proof test time stamps connected to the output *q_adtProofTests* (to confirm commissioning and first proof test of the XPSU safety module).<br>• The value at the output *q_ datTestStart* is updated. |
| *i_xTestExecuted* | BOOL | A rising edge at this input confirms that a proof test has been performed.<br><br>Results:<br>• A time stamp is added to the first array element of the array with the proof test time stamps connected to the output *q_adtProofTests*. The existing time stamps are moved by one position in the array (first in, first out (FIFO))<br>• The due date of the next proof test is provided at the output *q_ datNextProof*. |
| *i_xSaveBProofTests* | BOOL | A rising edge at this input triggers reading of the time stamps from the proof test array connected to the input *i_adtProofTests*. Refer to the description of the input *i_adtProofTests* for details. |
| *i_adtProofTests* | ARRAY[1..10] OF DATE_ AND_TIME | Input for connecting an external proof test array containing proof test time stamps which originate, for example, from an HMI application. |

| Output | Data type | Description |
|---|---|---|
| *q_xActive* | BOOL | If the value at this output is TRUE, the function block is being executed. |
| *q_xError* | BOOL | If the value at this output isTRUE, the function block has detected an error. Refer to *q_wErrorId* for details. |
| *q_wErrorId* | WORD | Provides information on detected errors. Error codes are listed in the section *Error Codes*, page 26. |
| *q_datTestStart* | DATE | Date value of the first proof test after activation of a new XPSU safety module via the input *i_ xNewModuleActive*. |
| *q_datNextProof* | DATE | Due date for the next proof test. |
| *q_xTestIntervViol* | BOOL | If the value at this output is TRUE, the proof test interval has been exceeded. |
| *q_xOpExceeded* | BOOL | If the value at this output is TRUE, the number of remaining cycles is 0 for at least one of the monitored safety-related inputs or safety-related outputs, or for the XPSU safety module itself. |
| *q_wExceededId* | WORD | ID of the safety-related input, or the safety-related output, or the XPSU safety module itself that has exceeded |

| Output | Data type | Description |
|---|---|---|
| | | the maximum number of cycles.<br>Format:<br>• For inputs: `16#101n`<br>• For outputs: `16#102n`<br>• For module: `16#100n`<br>Refer to *Output q_wExceededId*, page 26 for details. |
| *q_adtProofTests* | ARRAY[1..10] OF DATE_AND_TIME | Array of time stamps of the last ten proof tests performed. The first array element contains the latest time stamp (FIFO). |
| *q_udiNumOpSystem* | UDINT | Number of cycles of the XPSU safety module itself. The value is reset to zero if a new XPSU safety module is activated via the input *i_xNewModuleActive*. |
| *q_stRemainNumOp* | ST_RemainNumOp | Number of remaining cycles for each monitored safety-related input, for each safety-related output, and for the XPSU safety module itself, written to the structure *ST_RemainNumOp*. |

## Error Codes

If the function block detects an error, the output *q_xError* is set to TRUE and an error code is provided at the output *q_wErrorId*.

| Error code | Description |
|---|---|
| `16#1003` | The proof test interval value at the input *i_udiTestInterv* is invalid. The value must be greater than zero. |
| `16#1004` | A new XPSU safety module has not yet been activated via the input *i_xNewModuleActive* (no first proof test confirmed). The error code is also generated when the function block is started for the first time via the input *i_xEnable*. |
| `16#105n` | Invalid configuration of safety-related input in the structure ST_InputControl. “n” is a number from 1 to 6 which indicates the number of the affected structure (corresponds to the affected safety-related input). Refer to *ST_InputControl*, page 32 for details on the parameters. |

## Output q_wExceededId

If no cycles remain for a safety-related input, or for a safety-related output, or for the XPSU safety module itself, the value at the output *q_xOpExceeded* is set to TRUE. The value at the output *q_wExceededId* identifies the affected safety-related input, or safety-related output, or the XPSU safety module itself.

| ID | Description |
|---|---|
| `16#1005` | Remaining number of cycles for the XPSU safety module itself is 0. |
| `16#1011` | Remaining number of cycles for the safety-related input defined in the first element of the structure *ST_InputControl* is 0. |
| `16#1012` | Remaining number of cycles for the safety-related input defined in the second element of the structure *ST_InputControl* is 0. |
| `16#1013` | Remaining number of cycles for the safety-related input defined in the third element of the structure *ST_InputControl* is 0. |
| `16#1014` | Remaining number of cycles for the safety-related input defined in the fourth element of the structure *ST_InputControl* is 0. |
| `16#1015` | Remaining number of cycles for the safety-related input defined in the fifth element of the structure *ST_InputControl* is 0. |
| `16#1016` | Remaining number of cycles for the safety-related input defined in the sixth element of the structure *ST_InputControl* is 0. |

| ID | Description |
|---|---|
| `16#1021` | Remaining number of cycles for the safety-related output defined in the first element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1022` | Remaining number of cycles for the safety-related output defined in the second element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1023` | Remaining number of cycles for the safety-related output defined in the third element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1024` | Remaining number of cycles for the safety-related output defined in the fourth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1025` | Remaining number of cycles for the safety-related output defined in the fifth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1026` | Remaining number of cycles for the safety-related output defined in the sixth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1027` | Remaining number of cycles for the safety-related output defined in the seventh element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1028` | Remaining number of cycles for the safety-related output defined in the eighth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#1029` | Remaining number of cycles for the safety-related output defined in the ninth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#102A` | Remaining number of cycles for the safety-related output defined in the tenth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#102B` | Remaining number of cycles for the safety-related output defined in the eleventh element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#102C` | Remaining number of cycles for the safety-related output defined in the twelfth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |
| `16#102D` | Remaining number of cycles for the safety-related output defined in the thirteenth element of the structure *ST_DevControl* connected to the input *i_astControlOut* is 0. |

## Persistent Variables

Certain variables for the function block need to be declared as persistent variables so that they are not re-initialized after a controller restart.

Procedure for adding the instance paths of the contained persistent values to the list of persistent variables:

| Step | Action |
|---|---|
| 1 | In the Application tree under the Application node, add the **[Persistent Variables]** object. |
| 2 | Open the editor, right-click and select *Add all instance paths*. |

This adds the following persistent global variables to the list:

| Variable | Data IDs Output at *q_wExceededIdType* | Default Value | Description |
|---|---|---|---|
| *R_udiCountSystem* | UDINT | `0` | Total number of cycles of the XPSU safety module itself. |
| *R_audiCountInp* | ARRAY[1..6] OF UDINT | `0` (1) | Remaining number of cycles of the safety-related inputs.<br><br>The array size of six covers the maximum number of safety- |

| Variable | Data IDs Output at *q_wExceededIdType* | Default Value | Description |
|---|---|---|---|
| | | | related inputs that an XPSU safety module can provide. |
| *R_udiCountProc* | UDINT | `0` | Remaining number of cycles of the XPSU safety module itself. |
| *R_audiCountOut* | ARRAY[1..13] OF UDINT | `0` (1) | Remaining number of cycles of the safety-related outputs.<br>The array size of 13 covers the maximum number of safety-related outputs that an XPSU safety module with an XPSUEP extension module can provide. |
| *R_adtProof* | ARRAY[1..10] OF DATE_AND_TIME | `DT#1970-1-1-0:0:0.0` (1) | Array for the time stamps of the last ten proof tests (first in, first out). |
| *R_datTestStart* | DATE | `D#1970-1-1` | Time stamp of the first proof test performed for a new XPSU safety module after confirmation via the input *i_xNewModuleActive*. |

(1) The default value refers to each element of the array

The input *i_xNewModuleActive* of the function block *FB_XpsuMain* lets you reset *R_udiCountProc* to zero if an XPSU safety module is installed. The other counters are reset at the inputs of the function blocks via the corresponding reset bits in the structure elements.

# Structures

## *ST_DevControl* - General Information

### Overview

| | |
|---|---|
| Type: | Structure |
| Available as of: | V1.0.4.0 |
| Inherits from: | - |

### Description

The structure *ST_DevControl* is used for the cycle counter for the XPSU safety module itself and for the cycle counter for the safety-related outputs of the XPSU safety module. The number of cycles of a safety-related output allows you to determine the number of cycles of the equipment (actuator) connected to this safety-related output.

The structure is connected to the inputs *i_astControlOut* and *i_stControlProc* of the function block *FB_XpsuMain*.

The structure consists of 13 elements. The first seven elements correspond to the maximum number of safety-related outputs an XPSU safety module can have. The remaining six elements correspond to the number of safety-related outputs an XPSUEP extension module has.

### Structure Elements

| Name | Data type | Description |
|---|---|---|
| *udiMaxNumOp* | UDINT | Maximum number of cycles of the safety-related outputs of the XPSU safety module or maximum number of cycles of the XPSU safety module itself.<br><br>If the value is set to 0, the cycles are not counted. |
| *xReset* | BOOL | If the value of this parameter is set to TRUE, the cycle counter for the corresponding safety-related output is reset to 0.<br><br>Such a reset may be required if, for example, you have replaced equipment connected to a safety-related output.<br><br>If the maximum number of cycles of the safety-related output is reached (and after you have replaced affected equipment), a reset is required for the counting to resume.<br><br>**NOTE:** After resetting the counter by setting *xReset* to TRUE, *xReset* must be set back to FALSE by the application program. If the value remains TRUE, the counter is continuously reset. |

### Used By

- *FB_XpsuMain*

## *ST_DiagCodes* - General Information

### Overview

| | |
|---|---|
| Type: | Structure |
| Available as of: | V1.0.4.0 |
| Inherits from: | - |

### Description

The structure *ST_DiagCodes* contains the diagnostics codes, page 19 of the XPSU safety module, each represented as a BOOLEAN structure member. The structure is used at the output *q_stDiagCode* of the function block *FB_XpsuDiag*. Depending on the XPSU safety module used and on the bit sequence provided to the input *i_xDiagSignal* of the function block, the corresponding structure element is set to TRUE.

### Structure Elements

| **Name** | **Data type** | **Description** |
|---|---|---|
| *xOperational* | BOOL | Device in operating state Run, safety-related outputs activated. |
| *xWaitMonStartFTrig* | BOOL | Start input activated. Waiting for falling edge for monitored start. |
| *xWaitMonStartRTrig* | BOOL | Waiting for rising edge for automatic/manual or monitored start. |
| *xWaitStartUpTest* | BOOL | Waiting for startup test. |
| *xInp63ChangeState* | BOOL | Input S63 is expected to change its state. |
| *xInp62ChangeState* | BOOL | Input S62 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S62 and S63 are expected to change their state. |
| *xInp53ChangeState* | BOOL | Input S53 is expected to change its state. |
| *xInp52ChangeState* | BOOL | Input S52 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S52 and S53 are expected to change their state. |
| *xInp43ChangeState* | BOOL | Input S43 is expected to change its state. |
| *xInp42ChangeState* | BOOL | Input S42 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S42 and S43 are expected to change their state. |
| *xInp33ChangeState* | BOOL | Input S33 is expected to change its state. |
| *xInp32ChangeState* | BOOL | Input S32 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S32 and S33 are expected to change their state. |
| *xInp23ChangeState* | BOOL | Input S23 is expected to change its state. |
| *xInp22ChangeState* | BOOL | Input S22 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S22 and S23 are expected to change their state. |
| *xInp13ChangeState* | BOOL | Input S13 is expected to change its state. |
| *xInp12ChangeState* | BOOL | Input S12 is expected to change its state. In the case of a configuration with antivalent inputs, the inputs S12 and S13 are expected to change their state. |
| *xIoffAllNOCoff* | BOOL | Safety-related inputs deactivated, safety-related outputs deactivated. |
| *xIoffNOCoffDelayC* | BOOL | Instantaneous safety-related outputs are deactivated, delayed safety-related outputs are still activated. |

| Name | Data type | Description |
|---|---|---|
| *xSyncFltTestInRun* | BOOL | Synchronization alert. Both synchronized safety-related inputs have been activated, but not within the synchronization time. |
| *xSyncFltWaitTest* | BOOL | Synchronization alert. One of the synchronized safety-related inputs is still deactivated, but the synchronization time has already elapsed. |
| *xErrorS2x* | BOOL | Antivalence alert at input S2x. |
| *xErrorS1x* | BOOL | Antivalence alert at input S1x. |
| *xExtFltDelayCirc* | BOOL | Cross-circuit detected at input used for Cancel Delay function. |
| *xExtFltStartMonFdb* | BOOL | Cross-circuit detected at start input. |
| *xExtFltIn63Circ* | BOOL | Cross-circuit detected in circuit of input terminal S63. |
| *xExtFltIn62Circ* | BOOL | Cross-circuit detected in circuit of input terminal S62. |
| *xExtFltIn53Circ* | BOOL | Cross-circuit detected in circuit of input terminal S53. |
| *xExtFltIn52Circ* | BOOL | Cross-circuit detected in circuit of input terminal S52. |
| *xExtFltIn43Circ* | BOOL | Cross-circuit detected in circuit of input terminal S43. |
| *xExtFltIn42Circ* | BOOL | Cross-circuit detected in circuit of input terminal S42. |
| *xExtFltIn33Circ* | BOOL | Cross-circuit detected in circuit of input terminal S33. |
| *xExtFltIn32Circ* | BOOL | Cross-circuit detected in circuit of input terminal S32. |
| *xExtFltIn23Circ* | BOOL | Cross-circuit detected in circuit of input terminal S23. |
| *xExtFltIn22Circ* | BOOL | Cross-circuit detected in circuit of input terminal S22. |
| *xExtFltIn13Circ* | BOOL | Cross-circuit detected in circuit of input terminal S13. |
| *xExtFltIn12Circ* | BOOL | Cross-circuit detected in circuit of input terminal S12. |
| *xErrorConfig* | BOOL | Configuration error detected. |
| *xIntFltInExModule* | BOOL | General error detected in expansion module. |
| *xIntFlt* | BOOL | General error detected. |
| *xSupplyOutOfTol* | BOOL | Supply voltage out of tolerance. If the supply voltage drops below the lower limit value, the output Z1 is no longer supplied. |
| *xOpenWireL1L2* | BOOL | Wiring in circuit for voltage U12 interrupted (between L1 and L2, wire break). |
| *xOpenWireL3L2* | BOOL | Wiring in circuit for voltage U32 interrupted (between L3 and L2, wire break). |
| *xVoltOnAllNOCoff* | BOOL | Voltage at safety-related input is above the adjusted voltage threshold, device is in defined safe state. |
| *xL1L2AboveThres-hold* | BOOL | Voltage U12 does not meet the requirements for detected standstill while U32 already does. |
| *xL3L2AboveThres-hold* | BOOL | Voltage U32 does not meet the requirements for detected standstill while U12 already does. |

## Used By

- *FB_XpsuDiag*

## *ST_InputControl* - General Information

### Overview

| | |
|---|---|
| Type: | Structure |
| Available as of: | V1.0.4.0 |
| Inherits from: | - |

### Description

The structure *ST_InputControl* configures the cycle counter for the safety-related inputs of an XPSU safety module. The number of cycles of a safety-related input allows you to determine the number of cycles of the equipment (sensor, command device) connected to this safety-related input.

An array of six structures is connected to the input *i_astControlInp* of the function block *FB_XpsuMain*.

Each structure configures the input channel or input channels of a safety-related input of the XPSU safety module.

**NOTE:** The cycle counter is available for digital safety-related inputs (Snn), not for analog safety-related inputs (Lnn).

### Structure Elements

| Name | Data type | Description |
|---|---|---|
| *udiMaxNumOp* | UDINT | Maximum number of cycles of safety–related input specified with *byMonitorInput1* (and with *byMonitorInput2* in the case of XPSUDN or XPSUS safety modules which feature safety-related inputs having two input channels).<br><br>If the value is set to 0, the cycles are not counted. |
| *byMonitorInput1* | BYTE | Specifies the number of the input channel Snn of the safety-related input of the XPSU safety module.<br><br>Format: Number nn of the input channel Snn of the safety-related input to configure.<br><br>Examples: 12 configures S12, 62 configures S62.<br><br>If your XPSU safety module has safety-related inputs with two input channels (XPSUDN, XPSUS), you can configure the corresponding second input channel via the structure element *byMonitorInput2*.<br><br>If the value is set to 0, the input channel is not monitored. |

| Name | Data type | Description |
|---|---|---|
| *byMonitorInput2* | BYTE | Specifies the number of the second input channel of the safety-related input of an XPSU safety module if your XPSU safety module has safety-related inputs with two input channels (XPSUDN, XPSUS).<br><br>Format: Number nn of the second input channel Snn of the safety-related input configured with *byMonitorInput1*.<br><br>Examples: 13 configures S13, 63 configures S63.<br><br>If the value is set to 0, the input channel is not monitored. |
| *xReset* | BOOL | If this value is set to TRUE, the cycle counter for the safety-related input or input channel corresponding to this array element is reset to the value of *udiMaxNumOp*.<br><br>Such a reset may be required if, for example, you have replaced equipment connected to the safety-related input.<br><br>If the maximum number of cycles of the safety-related input is reached (and after you have replaced affected equipment), a reset is required for the counting to resume.<br>**NOTE:** After resetting the counter by setting *xReset* to TRUE, *xReset* must be set back to FALSE by the application program. If the value remains TRUE, the counter is continuously reset. |

## Used By

- *FB_XpsuMain*

# *ST_RemainNumOp* - General Information

## Overview

| Type: | Structure |
|---|---|
| Available as of: | V1.0.4.0 |
| Inherits from: | - |

## Description

The structure *ST_RemainNumOp* contains the remaining number of cycles of the safety-related inputs, the safety-related outputs, and the XPSU safety module itself. It is connected to the output *q_stRemainNumOp* of the function block *FB_XpsuMain*.

## Structure Elements

| **Name** | **Data type** | **Description** |
|---|---|---|
| *udiNumRemainingProc* | UDINT | Remaining number of cycles of the XPSU safety module itself. |
| *audiNumOpRemainInp* | ARRAY(1..6) OF UDINT | Remaining number of cycles of the safety-related inputs of the XPSU safety module.<br><br>Each array element contains the information for one safety-related input. |
| *audiNumOpRemainOut* | ARRAY(1..13) OF UDINT | Remaining number of cycles of the safety-related outputs of the XPSU safety module.<br><br>Each array element contains the information for one safety-related output (up to seven for the XPSU safety module plus an additional six for a connected XPSUEP extension module). |

## Used By

- *FB_XpsuMain*

# Index

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

As standards, specifications, and design change from time to time, please ask for confirmation of the information given in this publication.

© 2021 – Schneider Electric. All rights reserved.

EIO0000004435.00