Modicon X80
BMXEHC0200 Counting Module
User Manual

Original instructions

10/2019
The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an “as is” basis, at your own risk. All other rights are reserved.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2019 Schneider Electric. All rights reserved.
# Table of Contents

Safety Information .................................................. 7
About the Book .................................................... 11

## Part I Introduction to the Counting Function

### Chapter 1 General Information on the Counting Function
- General Information on Counting Functions .................. 17

### Chapter 2 Presentation of Counting Module
- General Information about Counting Module .................. 19
- General Information about the Counting Module Operation .......................... 21
- Presentation of the BMX EHC 0200 Counting Module ........ 22

### Chapter 3 Presentation of the Counting Module Operation
- Overview of BMX EHC 0200 Module Functionalities .......... 23

## Part II Counting Module BMX EHC 0200 Hardware Implementation

### Chapter 4 General Rules for Installing Counting Module BMX EHC 0200
- Physical Description of the Counting Module ................. 27
- Fitting of Counting Modules .................................. 28
- Fitting 10-Pin and 16-Pin Terminal Blocks to a BMX EHC 0200 Counting Module .................................. 30
- How to Connect BMX EHC 0200 Module: Connecting 16-Pin and 10-Pin Terminal Blocks .................................. 32

### Chapter 5 BMX EHC 0200 Counting Module Hardware Implementation
- Standards and Certifications .................................. 35
- Characteristics for the BMX EHC 0200 Module and its Inputs and Outputs ............................................. 37
- Display and Diagnostics of the BMX EHC 0200 Counting Module .......................... 40
- BMX EHC 0200 Module Wiring .................................. 42
- Shielding Connection Kit ....................................... 50

## Part III Counting Module BMX EHC 0200 Functionalities

### Chapter 6 BMX EHC 0200 Counting Module Functionalities
- BMX EHC 0200 Module Configuration .......................... 55
- Input Interface Blocks ......................................... 56
- Programmable Filtering ....................................... 57
- Comparison .................................................. 58

Characteristics for the BMX EHC 0200 Module and its Inputs and Outputs ............................................. 37
Chapter 10 BMX EHC xxxx Counting Module Settings ................................. 143
  Adjust Screen for BMX EHC 0200 Counting Modules .......................... 144
  Setting the Preset Value ....................................................... 146
  Setting the Calibration Factor ............................................. 147
  Modulo Adjust ..................................................................... 148
  Setting the Hysteresis Value ................................................. 149

Chapter 11 Debugging the BMX EHC 0200 Counting Modules ............... 151
  11.1 Debug Screen for BMX EHC xxxx Counting Modules ................. 152
  11.2 BMX EHC 0200 Module Debugging ..................................... 155
    Frequency Mode Debugging .................................................. 156
    Event Counting Mode Debugging ....................................... 157
    Period Measuring Mode Debugging .................................... 158
    Ratio Mode Debugging ...................................................... 159
    One Shot Counter Mode Debugging .................................... 160
    Modulo Loop Counter Mode Debugging ............................... 161
    Free Large Counter Mode Debugging ................................ 163
    Pulse Width Modulation Mode Debugging ............................ 165

Chapter 12 Display of BMX EHC xxxx Counting Module Error ............. 167
  Fault Display Screen for BMX EHC 0200 Counting Modules ............ 168
  Faults Diagnostics Display .................................................... 170
  List of Error ................................................................. 171

Chapter 13 The Language Objects of the Counting Function ............ 173
  13.1 The Language Objects and IODDT of the Counting Function ...... 174
    Introducing Language Objects for Application-Specific Counting ... 175
    Implicit Exchange Language Objects Associated with the Application- Specific Function .............................. 176
    Explicit Exchange Language Objects Associated with the Application- Specific Function ................................ 177
    Management of Exchanges and Reports with Explicit Objects .... 179
  13.2 Language Objects and IODDT Associated with the Counting Function of the BMX EHC xxxx Modules ............................. 184
    Details of Implicit Exchange Objects for the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs .... 185
    Details of the Explicit Exchange Objects for the T_CPT_BMX-type IODDT ................................................. 190
  13.3 The IODDT Type T_GEN_MOD Applicable to All Modules .......... 192
    Details of the Language Objects of the IODDT of Type T_GEN_MOD 192
| 13.4 | Device DDTs Associated with the Counting Function of the BMX EHC xxxx Modules | 194 |
|  | Counter Device DDT | 195 |
|  | MOD_FLT Byte Description | 203 |

**Part V** Quick Start: Example of Counting Module Implementation | 205 |

**Chapter 14** Description of the Application | 207 |

Overview of the Application | 207 |

**Chapter 15** Installing the Application Using Control Expert | 209 |

15.1 Presentation of the Solution Used | 210 |

Technological Choices Used | 211 |

Process Using Control Expert | 212 |

15.2 Developing the Application | 213 |

Creating the Project | 214 |

Configuration of the Counting Module | 215 |

Declaration of Variables | 218 |

Creating the Program for Managing the Counter Module | 220 |

Creating the Labelling Program in ST | 222 |

Creating the I/O Event Section in ST | 224 |

Creating a Program in LD for Application Execution | 225 |

Creating an Animation Table | 228 |

Creating the Operator Screen | 229 |

**Chapter 16** Starting the Application | 231 |

Execution of Application in Standard Mode | 231 |

**Index** | 235 |
Safety Information

Important Information

NOTICE

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 be made and that enough time is allowed to perform complete and satisfactory testing.

<table>
<thead>
<tr>
<th>❱ WARNING ❱</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQUIPMENT OPERATION HAZARD</td>
</tr>
<tr>
<td>● Verify that all installation and set up procedures have been completed.</td>
</tr>
<tr>
<td>● Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.</td>
</tr>
<tr>
<td>● Remove tools, meters, and debris from equipment.</td>
</tr>
<tr>
<td><strong>Failure to follow these instructions can result in death, serious injury, or equipment damage.</strong></td>
</tr>
</tbody>
</table>

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

At a Glance

Document Scope
This manual describes the hardware and software implementation of the Modicon X80 counting module BMXEHC0200.

Validity Note
This documentation is valid for EcoStruxure™ Control Expert 14.1 or later.

The technical characteristics of the devices described in the present document also appear online.
To access the information online:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Go to the Schneider Electric home page <a href="http://www.schneider-electric.com">www.schneider-electric.com</a>.</td>
</tr>
</tbody>
</table>
| 2    | In the Search box type the reference of a product or the name of a product range.  
  ● Do not include blank spaces in the reference or product range.  
  ● To get information on grouping similar modules, use asterisks (*). |
| 3    | If you entered a reference, go to the Product Datasheets search results and click on the reference that interests you.  
  If you entered the name of a product range, go to the Product Ranges search results and click on the product range that interests you. |
| 4    | If more than one reference appears in the Products search results, click on the reference that interests you. |
| 5    | Depending on the size of your screen, you may need to scroll down to see the datasheet. |
| 6    | To save or print a datasheet as a .pdf file, click Download XXX product datasheet. |

The characteristics that are presented 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

<table>
<thead>
<tr>
<th>Title of documentation</th>
<th>Reference number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical installation guide</td>
<td>EIGED306001EN (English)</td>
</tr>
<tr>
<td>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</td>
<td>EIO0000002726 (English), EIO0000002727 (French), EIO0000002728 (German), EIO0000002730 (Italian), EIO0000002729 (Spanish), EIO0000002731 (Chinese)</td>
</tr>
<tr>
<td>EcoStruxure™ Control Expert, Program Languages and Structure, Reference Manual</td>
<td>35006144 (English), 35006145 (French), 35006146 (German), 35013361 (Italian), 35006147 (Spanish), 35013362 (Chinese)</td>
</tr>
<tr>
<td>EcoStruxure™ Control Expert, Operating Modes</td>
<td>33003101 (English), 33003102 (French), 33003103 (German), 33003104 (Spanish), 33003696 (Italian), 33003697 (Chinese)</td>
</tr>
<tr>
<td>EcoStruxure™ Control Expert, I/O Management, Block Library</td>
<td>33002531 (English), 33002532 (French), 33002533 (German), 33002684 (Italian), 33002534 (Spanish), 33003685 (Chinese)</td>
</tr>
<tr>
<td>EcoStruxure™ Control Expert, Communication, Block Library</td>
<td>33002527 (English), 33002528 (French), 33002529 (German), 33002682 (Italian), 33002530 (Spanish), 33003683 (Chinese)</td>
</tr>
</tbody>
</table>

You can download these technical publications and other technical information from our website at [www.schneider-electric.com/en/download](http://www.schneider-electric.com/en/download).
Product Related Information

⚠️ WARNING

UNINTENDED EQUIPMENT OPERATION
The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program, install, alter, and apply this product.
Follow all local and national safety codes and standards.

Failure to follow these instructions can result in death, serious injury, or equipment damage.
Part I
Introduction to the Counting Function

Subject of this Part
This part provides a general introduction to the counting function and the operating principles of the BMX EHC 0200.

What Is in This Part?
This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General Information on the Counting Function</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>Presentation of Counting Module</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Presentation of the Counting Module Operation</td>
<td>23</td>
</tr>
</tbody>
</table>
Chapter 1
General Information on the Counting Function

General Information on Counting Functions

At a Glance

The counting function enables fast counting using couplers, Control Expert screens and specialized language objects. The general operation of expert modules also known as couplers is described in the section Presentation of the Counting Module Operation BMX EHC 0200.

In order to implement the counting, it is necessary to define the physical context in which it is to be executed (rack, supply, processor, modules etc.) and to ensure the software implementation (see page 109).

This second aspect is performed from the different Control Expert editors:

- in offline mode
- in online mode
Chapter 2
Presentation of Counting Module

Subject of this Chapter
This chapter deals with the Modicon X80 counting module BMX EHC 0200.

What Is in This Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Information about Counting Module</td>
<td>20</td>
</tr>
<tr>
<td>General Information about the Counting Module Operation</td>
<td>21</td>
</tr>
<tr>
<td>Presentation of the BMX EHC 0200 Counting Module</td>
<td>22</td>
</tr>
</tbody>
</table>
General Information about Counting Module

Introduction
Counting module is standard format module that enable pulses from a sensor to be counted at a maximum frequency of 60 KHz (BMX EHC 0200).
The BMX EHC 0200 module has 2 channels.

Sensors Used
The sensors used on each channel may be:

- 24 VDC two-wire proximity sensors
- Incremental signal encoders with 10/30 VDC output and push-pull outputs.

Illustration

1 Incremental encoder
2 Proximity sensors
3 Counting module BMX EHC 0200
General Information about the Counting Module Operation

Introduction
The BMX EHC 0200 module have:
- Counting-related functions (comparison, capture, homing, reset to 0)
- Event generation functions designed for the application program
- Outputs for actuator use (contacts, alarms, relays)

Characteristics
The main characteristics of BMX EHC 0200 module are as follows.

<table>
<thead>
<tr>
<th>Application</th>
<th>Number of channels per module</th>
<th>Number of physical inputs per channel</th>
<th>Number of physical outputs per channel</th>
<th>Maximum frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>60 KHz</td>
</tr>
<tr>
<td>Downcounting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up/Down counting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency meter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency generator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Axis monitoring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Presentation of the BMX EHC 0200 Counting Module

At a Glance

The BMX EHC 0200 counting module enables the counting or downcounting of pulses to be performed. It has the following functions:

- Enable
- Capture
- Comparison
- Homing or reset to 0
- 2 physical outputs

Structure of a counter channel

The following illustration shows the overall structure of a counter channel:
Overview of BMX EHC 0200 Module Functionalities

At a Glance
This part presents the different types of user applications for the BMX EHC 0200 module.

Measurement
The following table presents the measurement functionality for the BMX EHC 0200 module:

<table>
<thead>
<tr>
<th>User application type</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed measurement/stream measurement</td>
<td>Frequency</td>
</tr>
<tr>
<td>Random events monitoring</td>
<td>Event counting</td>
</tr>
<tr>
<td>Pulse evaluation/Speed control</td>
<td>Period measuring</td>
</tr>
<tr>
<td>Flow control</td>
<td>Ratio</td>
</tr>
</tbody>
</table>

Counting
The following table presents the counting functionality for the BMX EHC 0200 module:

<table>
<thead>
<tr>
<th>User application type</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping</td>
<td>One shot counter</td>
</tr>
<tr>
<td>Level 1 packaging/labeling</td>
<td>Modulo loop counter</td>
</tr>
<tr>
<td>Level 2 packaging/labeling</td>
<td>Free large counter</td>
</tr>
<tr>
<td>Accumulator</td>
<td>Free large counter</td>
</tr>
<tr>
<td>Axis control</td>
<td>Free large counter</td>
</tr>
</tbody>
</table>

NOTE: In case of a user application such as level 1 packaging/labeling, the machine makes constant spacing between parts. In case of a user application such as level 2 packaging/labeling, the counting module learns the incoming edge of each part.

Frequency Generator
The following table presents the frequency generator functionality for the BMX EHC 0200 module:

<table>
<thead>
<tr>
<th>User application type</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input frequency device</td>
<td>Pulse width modulation</td>
</tr>
</tbody>
</table>
Interface

The BMX EHC 0200 module may be interfaced with the following components:

- mechanical switch
- 24 VDC two-wire proximity sensor
- 24 VDC three-wire proximity sensor
- 10/30 VDC encoder with push-pull outputs
Part II
Counting Module BMX EHC 0200 Hardware Implementation

Subject of this Part
This part presents the hardware implementation of the BMX EHC 0200 counting module.

What Is in This Part?
This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>General Rules for Installing Counting Module BMX EHC 0200</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>BMX EHC 0200 Counting Module Hardware Implementation</td>
<td>35</td>
</tr>
</tbody>
</table>
Chapter 4
General Rules for Installing Counting Module BMX EHC 0200

Subject of this Chapter
This chapter presents the general rules for installing counting module BMX EHC 0200.

What Is in This Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Description of the Counting Module</td>
<td>28</td>
</tr>
<tr>
<td>Fitting of Counting Modules</td>
<td>30</td>
</tr>
<tr>
<td>Fitting 10-Pin and 16-Pin Terminal Blocks to a BMX EHC 0200 Counting Module</td>
<td>32</td>
</tr>
<tr>
<td>How to Connect BMX EHC 0200 Module: Connecting 16-Pin and 10-Pin Terminal Blocks</td>
<td>33</td>
</tr>
</tbody>
</table>
Physical Description of the Counting Module

Illustration

The figure below present the counting module BMX EHC 0200:

![BMX EHC 0200 Diagram]

Physical Elements of the Modules

The table below presents the elements of the counting module BMX EHC 0200:

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Module state LEDs:</td>
</tr>
<tr>
<td></td>
<td>● State LEDs at module level</td>
</tr>
<tr>
<td></td>
<td>● State LEDs at channel level</td>
</tr>
<tr>
<td>2</td>
<td>16-pin connector to connect the counter 0 sensors</td>
</tr>
<tr>
<td>3</td>
<td>16-pin connector to connect the counter 1 sensors</td>
</tr>
<tr>
<td>4</td>
<td>10-pin connector to connect:</td>
</tr>
<tr>
<td></td>
<td>● Auxiliary outputs</td>
</tr>
<tr>
<td></td>
<td>● Sensor power supplies</td>
</tr>
</tbody>
</table>
Accessories

The BMX EHC 0200 module requires the use of the following accessories:

- Two 16-pin terminal blocks
- One 10-pin terminal block
- One BMXXSP•••• shielding connection kit (*see page 50*)

**NOTE:** The two 16-pin connectors and the 10-pin connector are available under the reference BMX XTS HSC 20.
Fitting of Counting Modules

At a Glance

The counting modules are powered by the rack bus. The modules may be handled without turning off power supply to the rack, without damage or disturbance to the PLC.

Fitting operations (installation, assembly, and disassembly) are described below.

Installation Precautions

The counting modules may be installed in any of the positions in the rack except for the first two (marked PS and 00) which are reserved for the rack power supply module (BMX CPS ••••) and the processor (BMX P34 ••••) respectively. Power is supplied by the bus at the bottom of the rack (3.3 V and 24 V).

Before installing a module, you must take off the protective cap from the module connector located on the rack.

<table>
<thead>
<tr>
<th>DANGER</th>
</tr>
</thead>
</table>

HAZARD OF ELECTRIC SHOCK

- Turn off all power to sensor and pre-actuator devices before connection of disconnection of the terminal block.
- Remove the terminal block before plugging / unplugging the module on the rack.

Failure to follow these instructions will result in death or serious injury.

Installation

The diagram below shows counting module BMX EHC 0200 mounted on the rack:
The following table describes the different elements which make up the assembly below:

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BMX EHC 0200 counting module</td>
</tr>
<tr>
<td>2</td>
<td>Standard rack</td>
</tr>
</tbody>
</table>

**Installing the Module on the Rack**

The following table shows the procedure for mounting the counting module in the rack:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Position the locating pins situated at the rear of the module (on the bottom part) in the corresponding slots in the rack.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong>: Before positioning the pins, make sure you have removed the protective cover.</td>
<td>Steps 1 and 2</td>
</tr>
<tr>
<td>2</td>
<td>Swivel the module towards the top of the rack so that the module sits flush with the back of the rack. It is now set in position.</td>
<td><img src="image1.png" alt="Illustration" /></td>
</tr>
<tr>
<td>3</td>
<td>Tighten the mounting screw to ensure that the module is held in place on the rack.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Tightening torque</strong>: 0.4...1.5 N•m (0.30...1.10 lbf-ft)</td>
<td><img src="image2.png" alt="Illustration" /></td>
</tr>
</tbody>
</table>
Fitting 10-Pin and 16-Pin Terminal Blocks to a BMX EHC 0200 Counting Module

At a Glance
BMX EHC 0200 counting modules with 10-pin and 16-pin terminal block connections require terminal blocks to be connected to the module. The fitting operations (assembly and disassembly) are described below.

Installing the 10-Pin and 16-Pin Terminal Blocks

**DANGER**

HAZARD OF ELECTRIC SHOCK
Turn off all power to sensor and pre-actuator devices before connection or disconnection of the terminal block.
Failure to follow these instructions will result in death or serious injury.

If two 16-pin terminal blocks are used, each can be plugged into the middle or the top connector of the module. Therefore, despite the indicators on the terminal blocks and module, it is possible to invert the two terminal blocks and thus create incorrect wiring.

**CAUTION**

UNEXPECTED BEHAVIOR OF APPLICATION
Do careful tests on the wiring before any connection of material (sensors, actuators) and before any application tests.
Failure to follow these instructions can result in injury or equipment damage.

The following table shows the procedure for assembling the 10-pin and 16-pin terminal blocks onto a BMX EHC 0200 counting module:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Plug the 10-pin terminal block into the bottom connector of the module.</td>
</tr>
<tr>
<td>2</td>
<td>Plug the 16-pin terminal block into the middle connector of the module if it is used.</td>
</tr>
<tr>
<td>3</td>
<td>Plug the 16-pin terminal block into the top connector of the module if it is used.</td>
</tr>
</tbody>
</table>

**NOTE:** The three module connectors have indicators which show the proper direction to use for terminal block installation.
How to Connect BMX EHC 0200 Module: Connecting 16-Pin and 10-Pin Terminal Blocks

At a Glance

The BMX EHC 0200 counting module uses the following terminal blocks:

- Two 16-pin terminal blocks for the inputs
- One 10-pin terminal block for supplies outputs

Description of the 10 and 16 Pin Terminal Blocks

The following table shows the characteristics of the BMX EHC 0200 terminal blocks:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of terminal block</td>
<td>Spring terminal blocks</td>
</tr>
<tr>
<td>Number of wires accommodated</td>
<td>1</td>
</tr>
<tr>
<td>Wire gauge accommodated Minimum</td>
<td>AWG 20 (0.5 mm²)</td>
</tr>
<tr>
<td>Maximum</td>
<td>AWG 18 (1 mm²)</td>
</tr>
<tr>
<td>Wiring constraints</td>
<td>To insert and remove wires from the connectors, use a flat-tipped screwdriver with a 2.5 mm wide and 0.4 mm thick blade. With the screwdriver, push the flexible plate down on the outside (the side closest to the corresponding receptacle) to open the round receptacle. A screwing (rotating) or bending motion is not required.</td>
</tr>
</tbody>
</table>

⚠️ DANGER

HAZARD OF ELECTRIC SHOCK

Turn off all power to sensor and pre-actuator devices before connection or disconnection of the terminal block.

Failure to follow these instructions will result in death or serious injury.
Chapter 5
BMX EHC 0200 Counting Module Hardware Implementation

Subject of this Chapter
This chapter deals with the hardware characteristics of the BMX EHC 0200 module.

What Is in This Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standards and Certifications</td>
<td>36</td>
</tr>
<tr>
<td>Characteristics for the BMX EHC 0200 Module and its Inputs and Outputs</td>
<td>37</td>
</tr>
<tr>
<td>Display and Diagnostics of the BMX EHC 0200 Counting Module</td>
<td>40</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Wiring</td>
<td>42</td>
</tr>
<tr>
<td>Shielding Connection Kit</td>
<td>50</td>
</tr>
</tbody>
</table>
Standards and Certifications

Download

Click the link that corresponds to your preferred language to download standards and certifications (PDF format) that apply to the modules in this product line:

<table>
<thead>
<tr>
<th>Title</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</td>
<td>English: EIO00000002726</td>
</tr>
<tr>
<td></td>
<td>French: EIO00000002727</td>
</tr>
<tr>
<td></td>
<td>German: EIO00000002728</td>
</tr>
<tr>
<td></td>
<td>Italian: EIO00000002730</td>
</tr>
<tr>
<td></td>
<td>Spanish: EIO00000002729</td>
</tr>
<tr>
<td></td>
<td>Chinese: EIO00000002731</td>
</tr>
</tbody>
</table>
Characteristics for the BMX EHC 0200 Module and its Inputs and Outputs

Ruggedized Version
The BMX EHC 0200H (hardened) equipment is the ruggedized version of the BMX EHC 0200 (standard) equipment. It can be used at extended temperatures and in harsh chemical environments.

For more information, refer to chapter Installation in More Severe Environments (see Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications).

Altitude Operating Conditions
The characteristics in the table below apply to the modules BMX EHC 0200 and BMX EHC 0200H for use at altitude up to 2000 m (6560 ft). When the modules operate above 2000 m (6560 ft), apply additional derating.

For detailed information, refer to chapter Operating and Storage Conditions (see Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications).

General Characteristics
This table presents the general characteristics for the BMX EHC 0200 and BMX EHC 0200H modules:

<table>
<thead>
<tr>
<th>Module type</th>
<th>2 counting channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating temperature</td>
<td>BMX EHC 0200 0...60 ºC (32...140 ºF)</td>
</tr>
<tr>
<td></td>
<td>BMX EHC 0200H -25...70 ºC (-13...158 °F)</td>
</tr>
<tr>
<td>Maximum frequency at counting inputs</td>
<td>60 kHz</td>
</tr>
<tr>
<td>Number of inputs/outputs per counting channel</td>
<td>Inputs 6 Type three 24 VDC inputs</td>
</tr>
<tr>
<td></td>
<td>Outputs Two 24 VDC outputs</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Sensor supply voltage 19.2...30 VDC</td>
</tr>
<tr>
<td>Module consumption</td>
<td>Does not take into account sensors or encoder consumption</td>
</tr>
<tr>
<td></td>
<td>⊗ All inputs OFF: Typical: 15 mA</td>
</tr>
<tr>
<td></td>
<td>⊗ All inputs ON: Typical: 75 mA</td>
</tr>
<tr>
<td>Actuator supply current</td>
<td>500 mA maximum per output</td>
</tr>
<tr>
<td></td>
<td>2 A per module</td>
</tr>
<tr>
<td>Power distribution to sensors</td>
<td>Yes with short-circuit and overload protection - typical 300 mA (short-circuit limited to 2.5 A)</td>
</tr>
<tr>
<td>Hot replacement</td>
<td>Yes, under the following conditions: The module may be removed and reinserted into its location while the rack is switched on, but the counter may have to be revalidated when it is reinserted into its base.</td>
</tr>
</tbody>
</table>
### Input Characteristics

This table presents the general characteristics of the input channels for the module:

<table>
<thead>
<tr>
<th>Number of inputs per channel</th>
<th>Six 24 VDC inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs: IN A, IN B, IN SYNC, IN EN, IN REF, IN CAP</td>
<td></td>
</tr>
<tr>
<td>Voltage</td>
<td>30 VDC maximum</td>
</tr>
<tr>
<td>At state 1</td>
<td>Voltage</td>
</tr>
<tr>
<td>Current</td>
<td>5 mA (up to 30 VDC)</td>
</tr>
<tr>
<td>At state 0</td>
<td>Voltage</td>
</tr>
<tr>
<td>Current</td>
<td>&lt;1.5 mA</td>
</tr>
<tr>
<td>Current at 11 VDC</td>
<td>&gt;2 mA</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Width</th>
<th>Module only</th>
<th>32 mm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On the rack</td>
<td>32 mm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Height</td>
<td>Module only</td>
<td>103.76 mm</td>
</tr>
<tr>
<td></td>
<td>On the rack</td>
<td>103.76 mm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Depth</td>
<td>Module only</td>
<td>92 mm</td>
</tr>
<tr>
<td></td>
<td>On the rack</td>
<td>104.5 mm</td>
<td></td>
</tr>
<tr>
<td>Encoder compliance</td>
<td>10...30 VDC incremental encoder model with push-pull at outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insulation voltage of the ground to the bus</td>
<td>1500 V RMS for 1 min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rack 24 V supply bus</td>
<td>Current for the 24 V bus</td>
<td>Typical: 40 mA</td>
<td></td>
</tr>
<tr>
<td>Rack 3 V supply bus</td>
<td>Current for the 3 V bus</td>
<td>Typical: 200 mA</td>
<td></td>
</tr>
<tr>
<td>Module Cycle Time</td>
<td>1 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**WARNING**

**OVERHEATING MODULE**

Do not operate the BMX EHC 0200H at 70°C (158°F) if the sensor power supply is greater than 26.4 V or less than 21.1 V.

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

This table presents the general characteristics of the output channels for the module:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of outputs per channel</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>source 24 VDC 0.5 A</td>
</tr>
<tr>
<td><strong>Voltage</strong></td>
<td>19.2...30 VCC</td>
</tr>
<tr>
<td><strong>Minimum load current</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Maximum load current</strong></td>
<td></td>
</tr>
<tr>
<td>Each point</td>
<td>0.5 A</td>
</tr>
<tr>
<td>Per module</td>
<td>2 A</td>
</tr>
<tr>
<td><strong>Leakage current at state 0</strong></td>
<td>0.1 mA maximum</td>
</tr>
<tr>
<td><strong>Voltage drop at state 1</strong></td>
<td>3 VDC maximum</td>
</tr>
<tr>
<td><strong>Output current short-circuit</strong></td>
<td></td>
</tr>
<tr>
<td>Each point</td>
<td>1.5 A maximum</td>
</tr>
<tr>
<td><strong>Maximum load capacity</strong></td>
<td>50 μF</td>
</tr>
<tr>
<td><strong>Short-circuit and overload</strong></td>
<td>Channel protection</td>
</tr>
<tr>
<td><strong>Polarity for each output channel</strong></td>
<td></td>
</tr>
<tr>
<td>By default</td>
<td>Normal logic on both channels</td>
</tr>
<tr>
<td>User configuration</td>
<td>Reverse logic for one or several channels</td>
</tr>
<tr>
<td><strong>Maximum inductive load</strong></td>
<td></td>
</tr>
<tr>
<td>The inductive load is calculated using the following formula:</td>
<td></td>
</tr>
<tr>
<td>[ L = 0.5 / I^2 \times F ]</td>
<td></td>
</tr>
<tr>
<td>The formula above uses the following parameters:</td>
<td></td>
</tr>
<tr>
<td>L: load inductance in Henry</td>
<td></td>
</tr>
<tr>
<td>I: load current in Amperes</td>
<td></td>
</tr>
<tr>
<td>F: switching frequency in Hertz</td>
<td></td>
</tr>
</tbody>
</table>
Display and Diagnostics of the BMX EHC 0200 Counting Module

At a Glance

The BMX EHC 0200 counting module has LEDs that enable the status of the module to be viewed:

- Module state LEDs: RUN, ERR, I/O
- State LEDs for inputs/outputs of each channel: IA, IB, IS, IE, IP, IC, Q0 and Q1.

Illustration

The following drawing shows the display screen of the BMX EHC 0200 module:

Fault Diagnostics

The following table presents the various module states according to the LED states:

<table>
<thead>
<tr>
<th>Module status</th>
<th>LED indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>The module is faulty or switched off</td>
<td>○</td>
</tr>
<tr>
<td>The module has a fault</td>
<td>★</td>
</tr>
<tr>
<td>The module is not configured</td>
<td>○</td>
</tr>
<tr>
<td>The module has lost communication</td>
<td>○</td>
</tr>
<tr>
<td>The sensors have a supply fault</td>
<td>○</td>
</tr>
<tr>
<td>The actuators have a supply fault</td>
<td>○</td>
</tr>
<tr>
<td>Short circuit on output Q0</td>
<td>○</td>
</tr>
<tr>
<td>Short circuit on output Q1</td>
<td>○</td>
</tr>
<tr>
<td>The channels are operational</td>
<td>○</td>
</tr>
<tr>
<td>The voltage is present at output Q0</td>
<td>○</td>
</tr>
<tr>
<td>The voltage is present at output Q1</td>
<td>○</td>
</tr>
<tr>
<td>The voltage is present at input IN_A</td>
<td>○</td>
</tr>
<tr>
<td>The voltage is present at input IN_B</td>
<td>○</td>
</tr>
</tbody>
</table>
The voltage is present at input IN_SYNC | ● | ● | ○ | ●
--- | --- | --- | --- | ---
The voltage is present at input IN_EN | ● | ○ | ○ | ●
--- | --- | --- | --- | ---
The voltage is present at input IN_REF | ● | ● | ○ | ●
--- | --- | --- | --- | ---
The voltage is present at input IN_CAP | ● | ● | ○ | ●
--- | --- | --- | --- | ---

Legend
- ● LED on
- ○ LED off
- □ LED flashing slowly
- △ LED flashing fast

An empty cell indicates that the state of the LED(s) is not taken into account.
BMX EHC 0200 Module Wiring

At a Glance

The BMX EHC 0200 counting module uses the following:

- Two 16-pin connectors for the inputs
- One 10-pin connector for the outputs

**NOTE:** The two 16-pin connectors and the 10-pin connector are sold separately and are available in the BMX XTS HSC 20 connection kit.

---

**Field Sensors**

The module has type 3 of IEC 61131 inputs that support signals from mechanical switching equipment such as:

- Contact relays
- Push-buttons
- Limit switch sensors
- Switches with 2 or 3 wires

The equipment must have the following characteristics:

- Voltage drop less than 8 V
- Minimum operating current less than or equal to 2 mA
- Maximum current in blocked state less than or equal to 1.5 mA

The module complies with most encoders that have a supply of 10...30 V and push-pull outputs.

**NOTE:** The module 24 V supply for sensors has thermal and short-circuit protection.
**Assignment of the 16-Pin Connector**

The figure below shows the physical location of the pin numbers for the 16-pin connector:

![Diagram of 16-pin connector]

The symbol and description of each pin are described in the following table:

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 7, 8</td>
<td>24V_SEN</td>
<td>24 VDC output for sensors supply</td>
</tr>
<tr>
<td>5, 6, 13, 14</td>
<td>GND_SEN</td>
<td>24 VDC output for sensors supply</td>
</tr>
<tr>
<td>15, 16</td>
<td>FE</td>
<td>Functional earth</td>
</tr>
<tr>
<td>3</td>
<td>IN_A</td>
<td>Input A</td>
</tr>
<tr>
<td>4</td>
<td>IN_SYNC</td>
<td>Synchronization input</td>
</tr>
<tr>
<td>9</td>
<td>IN_B</td>
<td>Input B</td>
</tr>
<tr>
<td>10</td>
<td>IN_EN</td>
<td>Enable input selected</td>
</tr>
<tr>
<td>11</td>
<td>IN_REF</td>
<td>Homing input</td>
</tr>
<tr>
<td>12</td>
<td>IN_CAP</td>
<td>Capture input</td>
</tr>
</tbody>
</table>
Sensor Connections

The example below shows sensors with applied to inputs IN_A and IN_B and equipment with applied to inputs IN_EN and IN_SYNC:

1. IN_A input
2. IN_B input
3. IN_SYNC input (synchronization input)
4. IN_EN input (enable input)
Encoder Connection

The example below shows an incremental encoder used for axis control and the three auxiliary inputs used especially for the 32-bit counter mode:

1. Encoder (inputs A, B and Z)
2. IN_REF input (homing input)
3. IN_EN input (enable input)
4. IN_CAP input (capture input)
Connecting Outputs and Output Supplies

The figure below shows the connection of supplies and actuators to the 10-pin connector:

1. 24 V supply for actuators
2. 24 V supply for sensors
3. Actuator for the Q0 output of counting channel 0
4. Actuator for the Q1 output of counting channel 0
5. Actuator for the Q0 output of counting channel 1
6. Actuator for the Q1 output of counting channel 1

Field Actuators

The Q0 and Q1 outputs are limited by a maximum current of 0.5 A.

**NOTE:** The Q0 and Q1 outputs have a thermal protection as well as short-circuit protection.
Assignment of the 10-Pin Connector

The figure below shows the physical location of the pin numbers for the 10-pin connector:

The symbol and description of each pin are described in the table below:

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24V_IN</td>
<td>24 VDC input for sensors supply</td>
</tr>
<tr>
<td>2</td>
<td>GND_IN</td>
<td>0 VDC input for sensors supply</td>
</tr>
<tr>
<td>5</td>
<td>Q0-1</td>
<td>Q1 output for counting channel 0</td>
</tr>
<tr>
<td>6</td>
<td>Q0-0</td>
<td>Q0 output for counting channel 0</td>
</tr>
<tr>
<td>7</td>
<td>Q1-1</td>
<td>Q1 output for counting channel 1</td>
</tr>
<tr>
<td>8</td>
<td>Q1-0</td>
<td>Q0 output for counting channel 1</td>
</tr>
<tr>
<td>9</td>
<td>24V_OUT</td>
<td>24 VDC input for actuators supply</td>
</tr>
<tr>
<td>10</td>
<td>GND_OUT</td>
<td>0 VDC input for actuators supply</td>
</tr>
</tbody>
</table>
Safety Instructions

Electromagnetic perturbations may cause the application to operate in an unexpected manner. Follow all local and national safety codes and standards.

**DANGER**

HAZARD OF ELECTRIC SHOCK

If you cannot prove that the end of a shielded cable is connected to the local ground, the cable must be considered as dangerous and personal protective equipment (PPE) must be worn.

Failure to follow these instructions will result in death or serious injury.

**WARNING**

UNEXPECTED EQUIPMENT OPERATION

Follow these instructions to reduce electromagnetic perturbations:

- Adapt the programmable filtering to the frequency applied at the inputs.
- Use a shielded cable (connected to the functional ground) connected to pins 15 and 16 of the connector when using an encoder or a fast detector.

In a highly disturbed environment:

- Use the BMXXSP**** shielding connection kit (see page 50) to connect the shielding without programmable filtering and
- Use a specific 24 VDC supply for inputs and a shielded cable for connecting the supply to the module.

Failure to follow these instructions can result in death, serious injury, or equipment damage.
The figure below shows the recommended circuit for high-noise environment using the shielding connection kit:

Improper fuse selection could result in damage to the module.

**NOTICE**

**MODULE DAMAGE**
Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies.

*Failure to follow these instructions can result in equipment damage.*
Shielding Connection Kit

Introduction
The BMXXSP... shielding connection kit allows to connect the cable shielding directly to the ground and not to the module shielding to help protect the system from electromagnetic perturbations.

Connect the shielding on the cordsets for connecting:
- Analog module,
- Counter module,
- Encoder interface module,
- Motion control module,
- An XBT console to the processor (via shielded USB cable).

Kit References
Each shielding connection kit includes the following components:
- A metal bar
- Two sub-bases

The reference is dependent on the number of slots on the Modicon X80 rack:

<table>
<thead>
<tr>
<th>Modicon X80 rack</th>
<th>Number of slots</th>
<th>Shielding Connection Kit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMXXBP0400(H)</td>
<td>4</td>
<td>BMXXSP0400</td>
</tr>
<tr>
<td>BMEXBP0400(H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMXXBP0600(H)</td>
<td>6</td>
<td>BMXXSP0600</td>
</tr>
<tr>
<td>BMEXBP0600(H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMXXBP0800(H)</td>
<td>8</td>
<td>BMXXSP0800</td>
</tr>
<tr>
<td>BMEXBP0800(H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMEXBP0602(H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMXXBP1200(H)</td>
<td>12</td>
<td>BMXXSP1200</td>
</tr>
<tr>
<td>BMEXBP1200(H)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMEXBP1002(H)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Clamping Rings
Use clamping rings to connect the shielding on cordsets to the metal bar of the kit.

NOTE: The clamping rings are not included in the shielding connection kit.

Depending on the cable diameter, the clamping rings are available under the following references:
- STBXSP3010: small rings for cables with cross-section 1.5...6 mm² (AWG16...10).
- STBXSP3020: large rings for cables with cross-section 5...11 mm² (AWG10...7).
Kit Installation

Installation of the shielding connection kit to the rack can be done with module already installed on the rack except for the BMXXBE0100 rack extender module.

Fasten the sub-bases of the kit at each end of the rack to provide a connection between the cable and the ground screw of the rack:

1 rack
2 sub-base
3 metallic bar
4 clamping ring

Tightening torques to install the shielding connection kit:
- For the screws fixing the sub-base to the Modicon X80 rack: Max. 0.5 N•m (0.37 lbf-ft)
- For the screws fixing the metallic bar to the sub-bases: Max. 0.75 N•m (0.55 lbf-ft)

NOTE: A shielding connection kit does not modify the volume required when installing and uninstalling modules.
Kit Dimensions

The following figure gives the dimensions (height and depth) of a Modicon X80 rack with its shielding connection kit:

NOTE: The overall width equals to the width of the Modicon X80 rack.
Part III
Counting Module BMX EHC 0200 Functionalities
Chapter 6
BMX EHC 0200 Counting Module Functionalities

Subject of this Chapter
This chapter deals with functionalities and counting modes of the BMX EHC 0200 module.

What is in This Chapter?
This chapter contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>BMX EHC 0200 Module Configuration</td>
<td>56</td>
</tr>
<tr>
<td>6.2</td>
<td>BMX EHC 0200 Module Operation Modes</td>
<td>82</td>
</tr>
</tbody>
</table>
Section 6.1
BMX EHC 0200 Module Configuration

Subject of this Section
This section deals with the configuration of the BMX EHC 0200 module.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Interface Blocks</td>
<td>57</td>
</tr>
<tr>
<td>Programmable Filtering</td>
<td>58</td>
</tr>
<tr>
<td>Comparison</td>
<td>59</td>
</tr>
<tr>
<td>Output Block Functions</td>
<td>62</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>67</td>
</tr>
<tr>
<td>Synchronization, Homing, Enable, Reset to 0 and Capture Functions</td>
<td>69</td>
</tr>
<tr>
<td>Modulo Flag and Synchronization Flag</td>
<td>77</td>
</tr>
<tr>
<td>Sending Counting Events to the Application</td>
<td>79</td>
</tr>
</tbody>
</table>
Input Interface Blocks

Description
The BMX EHC 0200 counting module has six inputs:

- 3 fast inputs
- 3 classic inputs

Fast Inputs
The table below presents the module’s fast inputs.

<table>
<thead>
<tr>
<th>Input</th>
<th>Use with sensors</th>
<th>Use with an encoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_A input</td>
<td>Clock input for measurement or single upcounting</td>
<td>For signal A</td>
</tr>
<tr>
<td>IN_B input</td>
<td>Second clock input for differential counting or measurement</td>
<td>For signal B</td>
</tr>
<tr>
<td>IN_SYNC input</td>
<td>Main synchronization input used for starting and homing</td>
<td>For signal Z</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for homing</td>
</tr>
</tbody>
</table>

Classic Inputs
The table below presents the module’s classic inputs:

<table>
<thead>
<tr>
<th>Input</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_EN input</td>
<td>Used to authorize counter operation</td>
</tr>
<tr>
<td>IN_REF input</td>
<td>Used for homing in advanced mode</td>
</tr>
<tr>
<td>IN_CAP input</td>
<td>Used for register capture</td>
</tr>
</tbody>
</table>
Programmable Filtering

At a Glance

The BMX EHC 0200 counting module’s six inputs are compatible with the use of mechanical switches.

A programmable debounce filter with 3 levels (low, medium and high) is available at every input.

Debounce Filter Diagram

The figure below shows the debounce filter with a low filtering level:

In this mode, the system delays all transitions until the signal is stable for 450 $\mu$s.

Selecting the Filtering Level

The table below specifies the characteristics of each input for the selected level of filtering:

<table>
<thead>
<tr>
<th>Filtering level</th>
<th>Input</th>
<th>Maximum delay</th>
<th>Minimum pulse</th>
<th>Maximum frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>IN_A, IN_B</td>
<td>-</td>
<td>5 $\mu$s</td>
<td>60 KHz</td>
</tr>
<tr>
<td></td>
<td>IN_SYNC</td>
<td>-</td>
<td>5 $\mu$s</td>
<td>200 Hz</td>
</tr>
<tr>
<td></td>
<td>IN_EN</td>
<td>50 $\mu$s</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>IN_CAP, IN_REF</td>
<td>-</td>
<td>50 $\mu$s</td>
<td>200 Hz</td>
</tr>
<tr>
<td>Low for bounces &gt; 2 KHz</td>
<td>IN_A, IN_B</td>
<td>-</td>
<td>450 $\mu$s</td>
<td>1 KHz</td>
</tr>
<tr>
<td></td>
<td>IN_EN</td>
<td>450 $\mu$s</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>IN_SYNC, IN_CAP, IN_REF</td>
<td>-</td>
<td>500 $\mu$s</td>
<td>200 Hz</td>
</tr>
<tr>
<td>Resource for bounces &gt; 1 KHz</td>
<td>IN_A, IN_B</td>
<td>-</td>
<td>1.25 ms</td>
<td>350 Hz</td>
</tr>
<tr>
<td></td>
<td>IN_EN</td>
<td>1.25 ms</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>IN_SYNC, IN_CAP, IN_REF</td>
<td>-</td>
<td>1.25 ms</td>
<td>200 Hz</td>
</tr>
<tr>
<td>High for bounces &gt; 250 Hz</td>
<td>IN_A, IN_B</td>
<td>-</td>
<td>4.2 ms</td>
<td>100 Hz</td>
</tr>
<tr>
<td></td>
<td>IN_EN</td>
<td>4.2 ms</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>IN_SYNC, IN_CAP, IN_REF</td>
<td>-</td>
<td>4.2 ms</td>
<td>100 Hz</td>
</tr>
</tbody>
</table>
Comparison

At a Glance
The comparison block operates automatically. This block is available in certain counting modes:
- Frequency
- Period measuring
- Ratio
- One shot counter
- Modulo loop counter
- Free large counter

Comparison Thresholds
The comparison block has two thresholds:
- The upper threshold: $\text{upper\_th\_value\ double word} (\%QDr.m.c.4)$
- The lower threshold: $\text{lower\_th\_value\ double word} (\%QDr.m.c.2)$

The upper threshold value must be greater than the lower threshold value.
If the upper threshold is less than or equal to the lower threshold, the lower threshold does not change but it is ignored.
This rule takes into account the format of the counter value.

Comparison Status Register
The result of the comparison is stored in the $\text{compare\_status\ register} (\%IWr.m.c.1)$.
The values of the two capture registers and the current value of the counter are compared with the thresholds.
The possible results are:
- Low: The value is less than the lower threshold value.
- Window: The value is between the upper and lower thresholds or equal to one of the two thresholds.
- High: The value is greater than the upper threshold.
The $\text{compare\_enable\ register} (\%IWr.m.c.1)$ consists of:

<table>
<thead>
<tr>
<th>Status register bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compared element</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comparison result</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>High</td>
<td>Window</td>
<td>Low</td>
<td>High</td>
<td>Window</td>
<td>Low</td>
<td>High</td>
<td>Window</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

35013355 10/2019
Update

When the `compare_enable` bit (%QWr.m.c.0.5) is set to 0, the comparison status register is deleted.

The comparison with capture 0 and capture 1 registers values is performed every time the registers are loaded.

The comparison with the counter current value is performed as follows:

<table>
<thead>
<tr>
<th>Counting mode</th>
<th>Registers updating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Intervals of 10 ms</td>
</tr>
<tr>
<td>Period measuring</td>
<td>At the end of the period</td>
</tr>
<tr>
<td>Ratio</td>
<td>Intervals of 10 ms</td>
</tr>
<tr>
<td>Event counting</td>
<td>Period intervals defined by the user</td>
</tr>
<tr>
<td>One shot counter</td>
<td>Intervals of 1 ms</td>
</tr>
<tr>
<td></td>
<td>Counter reloading</td>
</tr>
<tr>
<td></td>
<td>Counter stops</td>
</tr>
<tr>
<td></td>
<td>Threshold crossing</td>
</tr>
<tr>
<td>Modulo loop</td>
<td>Intervals of 1 ms</td>
</tr>
<tr>
<td></td>
<td>Counter reloading or resetting to 0</td>
</tr>
<tr>
<td></td>
<td>Counter stops</td>
</tr>
<tr>
<td></td>
<td>Threshold crossing</td>
</tr>
<tr>
<td>Free large counter</td>
<td>Intervals of 1 ms</td>
</tr>
<tr>
<td></td>
<td>Counter reloading</td>
</tr>
<tr>
<td></td>
<td>Threshold crossing</td>
</tr>
<tr>
<td>Pulse width modulation</td>
<td>Function not available in this mode</td>
</tr>
</tbody>
</table>
Modification of the Thresholds during the Operational Phase

When the `compare_enable` bit (%Qwr.m.c.0.5) is set to 0, the comparison status register is deleted.

When the `compare_suspend` bit (%Qwr.m.c.0.6) is set to 1, the value of the comparison status register is frozen until the bit switches back to 0.

The application may change threshold values without causing any disturbance when the `compare_suspend` bit (%Qwr.m.c.0.6) is set to 1.

This functionality allows modifying the application thresholds without modifying the status register behaviour.

When this bit switches back to 0, the comparisons restart with new threshold values.

The following figure illustrates the actions of the `compare_enable` bit (%Qwr.m.c.0.5) and the `compare_suspend` bit (%Qwr.m.c.0.6):

<table>
<thead>
<tr>
<th>compare_enable bit</th>
<th>compare_suspend bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>upper_th_value</th>
<th>lower_th_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( =X_0 )</td>
<td>( =Y_0 )</td>
</tr>
<tr>
<td>( =X_1 )</td>
<td>( =Y_1 )</td>
</tr>
<tr>
<td>( =X_2 )</td>
<td>( =Y_2 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>compare_status register</th>
</tr>
</thead>
<tbody>
<tr>
<td>According to ( X_0, Y_0 )</td>
</tr>
<tr>
<td>According to ( X_0, Y_0 ), Maintain</td>
</tr>
</tbody>
</table>
Output Block Functions

Output Function Blocks

Every channel in the counting module has two programmable output blocks that operate with the comparison status register and affect the behavior of physical outputs Q0 and Q1.

There are two ways to control the output:
- From the application: in this case, the output corresponds to the status of the output bit from the output command bit.
- From the output function block: in this case, the user must enable the output block function. Then, the output corresponds to the status of the output bit from the function block.

The following figure shows the output function block Q0:

Use of the Function Block

Every physical output is controlled by two bits:
- output_block_0_enable (%Q.m.c.2) and output_0 (%Q.m.c.0) for block 0
- output_block_1_enable (%Q.m.c.3) and output_1 (%Q.m.c.1) for block 1

The output_block_0(1)_enable bit enables the operation of the function block 0(1) to be authorized when it is set to 1. When the bit is set to 0, Bit output_block_0(1) is maintained at 0.

The output_0(1) bit is applied at the logic output Q0(1) and must be set to 0 when the function block is used. When the bit is set to 1, the output is forced to 1.

In the operational modes where the block generates a pulse, the pulse width can be configured thanks to the configuration screen.
Output Programming

The table below shows the configurable functions:

<table>
<thead>
<tr>
<th>Function code</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disabled = no direct action (Default value)</td>
</tr>
<tr>
<td>1</td>
<td>Low counter. The output is high if the counter value is less than the low threshold.</td>
</tr>
<tr>
<td>2</td>
<td>Counter in a window. The output is high if the counter value is between the upper and lower thresholds or equal to one of the two thresholds.</td>
</tr>
<tr>
<td>3</td>
<td>High counter. The output is high if the counter value is greater than the upper threshold.</td>
</tr>
<tr>
<td>4</td>
<td>Pulse less than the lower threshold. The output pulse starts when the counter value decreases and crosses the lower threshold value -1.</td>
</tr>
<tr>
<td>5</td>
<td>Pulse greater than the lower threshold. The output pulse starts when the counter value increases and crosses the lower threshold value +1.</td>
</tr>
<tr>
<td>6</td>
<td>Pulse less than the upper threshold. The output pulse starts when the counter value decreases and crosses the upper threshold value -1.</td>
</tr>
<tr>
<td>7</td>
<td>Pulse greater than the upper threshold. The output pulse starts when the counter value increases and crosses the upper threshold value +1.</td>
</tr>
<tr>
<td>8</td>
<td>Counter stopped (only in one shot counter mode). The output changes to high if the counter is stopped.</td>
</tr>
<tr>
<td>9</td>
<td>Counter running (only in one shot counter mode). The output changes to high if the counter is running.</td>
</tr>
<tr>
<td>10</td>
<td>Capture 0 low value. The output is high if the capture 0 value is less than the lower threshold.</td>
</tr>
<tr>
<td>11</td>
<td>Capture 0 value in a window. The output is high if the capture 0 value is between the upper and lower thresholds or equal to one of the two thresholds.</td>
</tr>
<tr>
<td>12</td>
<td>Capture 0 high value. The output is high if the capture 0 value is greater than the upper threshold.</td>
</tr>
<tr>
<td>13</td>
<td>Capture1 low value. The output is high if the capture1 value is less than the lower threshold.</td>
</tr>
</tbody>
</table>
NOTE: The output 0 function block is inactive when using the counter in pulse width modulation mode.

Output Performances
In general, these reflex actions act with a delay less than 0.6 ms. The repeatability is about +/- 0.3 ms.

Special boost functions:
- "Counter Low" (function code 1) applied to Output Block 0
- "Counter High" (function code 3) applied to Output Block 1 speed up timing.

Delay is less than 0.2 ms. The repeatability is about +/- 1 s.

Output Properties
The counting module BMX EHC 0200 enables output signals to be exchanged with two 24VCC field actuators.

It is possible to configure the following parameters for each output:
- The module response for fault recovery
- The output polarity for each counting channel (positive or negative polarity)
- The fallback mode and state for every module channel

These three parameters are described in the following pages.

<table>
<thead>
<tr>
<th>Function code</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Capture1 value in a window. The output is high if the capture1 value is between the upper and lower thresholds or equal to one of the two thresholds.</td>
</tr>
<tr>
<td>15</td>
<td>Capture1 high value. The output is high if the capture1 value is greater than the upper threshold.</td>
</tr>
</tbody>
</table>
Fault Recovery response

Outputs Q0 and Q1 are current limited (0.5 A maximum).

A thermal shutdown protects each output.

When a short-circuit is detected on one of the output channels, the counting module enables one of the two following actions according to the configuration:

- fault recovery parameter configured as latched off: The counting module latches off the output channel
- fault recovery parameter configured as autorecovery: The counting module latches off the output channel and automatically attempts to recover the error and to resume operation on the channel when the error is corrected.

In case of the fault recovery parameter is configured to latched off, if an output channel has been latched off because of short-circuit detection, the counting module recovers the fault upon the following sequence is processed:

- The error has been corrected
- You explicitly reset the fault: To reset the error, the application software must:
  - Reset the output_block_enable bit if it is active
  - Command the output to 0 V (depends on the polarity).

In case of the fault recovery parameter is configured to autorecovery, an output channel that has been turned off because of error detection starts operating again as soon as the error is corrected. No user intervention is required to reset the channels.

NOTE: A minimum delay of 10 s occurs before the error is cleared in both latched off and auto recovery modes.

Output Polarity Programming

It is possible to configure the polarity parameter for each output during the channel configuration:

- polarity parameter configured as polarity +: The physical output is 24 VDC when the output is at the high level (output_0_echo = 1)
- polarity parameter configured as polarity -: The physical output is 24 VDC when the output is at the low level (output_0_echo = 0)

By default, the two output channels are in positive polarity.
Output Fallback Modes

The fallback modes are the predefined states to which the output channels revert when the channel is not controlled by the processor (when communications are lost or when the processor is stopped for example).

The fallback mode of each output channel can be configured as one of the following modes:

- Fallback value: With. You may configure the fallback value to apply as 0 or 1
- Fallback value: Without. The output block function continues to operate according to the last received commands.

**NOTE:** By default, the fallback mode of the 2 output channels is with and the fallback value parameter is 0.
Diagnostics

Consistency Rules for Inputs Interface
The input interface requires that the sensor power supply remains active for counting operations. 
When the sensor power supply interrupts lasts 1 ms or less, the counter remains stable.
In case of power interrupt is greater than 1 ms, all counter values are disabled.
By default, the sensor supply fault makes the CH_ERROR (%Ir.m.c.ERR) global status bit at the high level and the red led IO lighted.
The configuration screen allows to unlink the sensor supply fault to the CH_ERROR bit by configuring the parameter Input Supply Fault as local instead of General IO Fault.

IODDT_VAR1 is of the type T_Unsigned_CPT_BMX or T_Signed_CPT_BMX

Consistency Rules for Outputs Interface
The output interface requires that the actuator power supply remains active for output blocks functions operations.
When the actuator supply voltage is insufficient the outputs are held to 0 V.
By default, the actuator supply fault makes the CH_ERROR (%Ir.m.c.ERR) global status bit at the high level and the red led IO lighted.
The configuration screen allows to unlink the actuator supply fault to the CH_ERROR bit by configuring the parameter Output Supply Fault as local instead of General IO Fault.

IODDT_VAR1 is of the type T_Unsigned_CPT_BMX or T_Signed_CPT_BMX

Explicit channel status words
The table below presents the composition of the %MNw.m.c.2 and %MNw.m.c.3 status words:

<table>
<thead>
<tr>
<th>Status Word</th>
<th>Bit position</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>%MNw.m.c.2</td>
<td>0</td>
<td>External fault at inputs</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>External fault at outputs</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Internal error or self-testing.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Configuration Fault</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Communication Error</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Application fault</td>
</tr>
<tr>
<td>%MNw.m.c.3</td>
<td>2</td>
<td>Sensor supply fault</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Actuator supply fault</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Short circuit on output Q0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Short circuit on output Q1</td>
</tr>
</tbody>
</table>
### IO Data

All input/output statuses are provided in the channel data bits.

The table below shows the channel data bits:

<table>
<thead>
<tr>
<th>Input/Output data field</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Ir.m.c.0</td>
<td>Logical state of output Q0</td>
</tr>
<tr>
<td>%Ir.m.c.1</td>
<td>Logical state of output Q1</td>
</tr>
<tr>
<td>%Ir.m.c.2</td>
<td>State of the output block function 0</td>
</tr>
<tr>
<td>%Ir.m.c.3</td>
<td>State of the output block function 1</td>
</tr>
<tr>
<td>%Ir.m.c.4</td>
<td>Electrical state of IN_A input</td>
</tr>
<tr>
<td>%Ir.m.c.5</td>
<td>Electrical state of IN_B input</td>
</tr>
<tr>
<td>%Ir.m.c.6</td>
<td>Electrical state of IN_SYNC input</td>
</tr>
<tr>
<td>%Ir.m.c.7</td>
<td>Electrical state of IN_EN input</td>
</tr>
<tr>
<td>%Ir.m.c.8</td>
<td>Electrical state of IN_REF input</td>
</tr>
<tr>
<td>%Ir.m.c.9</td>
<td>Electrical state of IN_CAP input</td>
</tr>
</tbody>
</table>
Synchronization, Homing, Enable, Reset to 0 and Capture Functions

Introduction
This section presents the functions used by the various counting modes of the BMX EHC 0200 module:

- Synchronization function
- Homing function
- Enable function
- Reset to 0 function
- Capture functions

Each function uses at least one of the following two bits:

- `valid_(function)` bit: Setting this bit to 1 allows you to take into account the occurrence of an external event which activates the function. If this bit is set to 0, the event is not taken into account and does not activate the function. The `functions_enabling` word (%QWr.m.c.0) contains all the `valid_(function)` bits.

- `force_(function)` bit: Setting this bit to 1 allows you to activate the function irrespective of the status of the external event. All the `force_(function)` bits are %Qr.m.c.4...%Qr.m.c.8 language objects.

Synchronization Function
The synchronization function is used to synchronize the counter operation upon a transition applied to the IN_SYNC (%I r.m.c.6) physical input or the `force_sync` bit set to 1.

This function is usable in the following counting modes:

- Pulse width modulation: to restart the output signal at the beginning (phase at 1)
- Modulo loop counter: to reset and start the counter
- One shot counter: to preset and start the counter
- Event counting: to restart the internal time base at the beginning

The user may configure the synchro edge parameter in the configuration screen by choosing from the following two possibilities to configure the sensitive edge that carries out the synchronization:

- Rising edge of the IN_SYNC input
- Falling edge of the IN_SYNC input
The following table presents the `force_sync` bit in bold which is an element of the `%Qr.m.c.d` output command word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Qr.m.c.0</td>
<td>OUTPUT_0</td>
<td>Forces OUTPUT_0 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.1</td>
<td>OUTPUT_1</td>
<td>Forces OUTPUT_1 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.2</td>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>Implementation of output 0 function block</td>
</tr>
<tr>
<td>%Qr.m.c.3</td>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>Implementation of output 1 function block</td>
</tr>
<tr>
<td>%Qr.m.c.4</td>
<td>FORCE_SYNC</td>
<td>Counting function synchronization and start</td>
</tr>
<tr>
<td>%Qr.m.c.5</td>
<td>FORCE_REF</td>
<td>Set to preset counter value</td>
</tr>
<tr>
<td>%Qr.m.c.6</td>
<td>FORCE_ENABLE</td>
<td>Implementation of counter</td>
</tr>
<tr>
<td>%Qr.m.c.7</td>
<td>FORCE_RESET</td>
<td>Reset counter</td>
</tr>
<tr>
<td>%Qr.m.c.8</td>
<td>SYNC_RESET</td>
<td>Reset SYNC_REF_FLAG</td>
</tr>
<tr>
<td>%Qr.m.c.9</td>
<td>MODULO_RESET</td>
<td>Reset MODULO_FLAG</td>
</tr>
</tbody>
</table>

The following table presents the `valid_sync` bit in bold which is an element of the `%Qwr.m.c.0` function enabling word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Qwr.m.c.0.0</td>
<td>VALID_SYNC</td>
<td>Synchronization and start authorization for the counting function via the IN_SYNC input</td>
</tr>
<tr>
<td>%Qwr.m.c.0.1</td>
<td>VALID_REF</td>
<td>Operation authorization for the internal preset function</td>
</tr>
<tr>
<td>%Qwr.m.c.0.2</td>
<td>VALID_ENABLE</td>
<td>Authorization of the counter enable via the IN_EN input</td>
</tr>
<tr>
<td>%Qwr.m.c.0.3</td>
<td>VALID_CAPT_0</td>
<td>Capture authorization in the capture0 register</td>
</tr>
<tr>
<td>%Qwr.m.c.0.4</td>
<td>VALID_CAPT_1</td>
<td>Capture authorization in the capture1 register</td>
</tr>
<tr>
<td>%Qwr.m.c.0.5</td>
<td>COMPARE_ENABLE</td>
<td>Comparators operation authorization</td>
</tr>
<tr>
<td>%Qwr.m.c.0.6</td>
<td>COMPARE_SUSPEND</td>
<td>Comparator frozen at its last value</td>
</tr>
</tbody>
</table>
The following table presents the synchronization principle:

<table>
<thead>
<tr>
<th>Edge Description</th>
<th>Status of the valid_sync (QWr.m.c.0.0) bit</th>
<th>Status of the counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rising or falling edge on IN_SYNC (depending on the configuration)</td>
<td>Set to 0</td>
<td>Not synchronized</td>
</tr>
<tr>
<td>Rising or falling edge on IN_SYNC (depending on the configuration)</td>
<td>Set to 1</td>
<td>Synchronized</td>
</tr>
<tr>
<td>Rising edge on force_sync (Qr.m.c.4) bit</td>
<td>Set to 0 or 1</td>
<td>Synchronized</td>
</tr>
</tbody>
</table>

When the synchronization occurs, the application can react using:

- either the SYNC_REF_FLAG input (%IWr.m.c.0.2) (see page 77)
- or the EVT_SYNC_PRESET input (%IWr.m.c.10.2) (see page 79).

**Homing Function**

This homing function loads the value predefined in the adjust screen preset value (%MDr.m.c.6) into the counter when the preset condition (defined by the preset mode parameter) occurs. This preset condition takes into account the IN_SYNC and IN_REF physical inputs to define the reference point of the process.

This function is only used in the free large counter mode.

The user may change the Preset Mode parameter in the configuration screen by choosing from the following five possibilities to configure the preset condition:

- Rising edge of the IN_SYNC input
- Rising edge of the IN_REF input
- Rising edge of the IN_SYNC input and high level of the IN_REF input
- First rising edge of the IN_SYNC input and high level of the IN_REF input
- First rising edge of the IN_SYNC input and low level of the IN_REF input
The following table presents the **force_ref** bit in bold which is an element of the %Qr.m.c.d output command word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Qr.m.c.0</td>
<td>OUTPUT_0</td>
<td>Forces OUTPUT_0 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.1</td>
<td>OUTPUT_1</td>
<td>Forces OUTPUT_1 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.2</td>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>Implementation of output 0 function block</td>
</tr>
<tr>
<td>%Qr.m.c.3</td>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>Implementation of output 1 function block</td>
</tr>
<tr>
<td>%Qr.m.c.4</td>
<td>FORCE_SYNC</td>
<td>Counting function synchronization and start</td>
</tr>
<tr>
<td>%Qr.m.c.5</td>
<td>FORCE_REF</td>
<td>Set to preset counter value</td>
</tr>
<tr>
<td>%Qr.m.c.6</td>
<td>FORCE_ENABLE</td>
<td>Implementation of counter</td>
</tr>
<tr>
<td>%Qr.m.c.7</td>
<td>FORCE_RESET</td>
<td>Reset counter</td>
</tr>
<tr>
<td>%Qr.m.c.8</td>
<td>SYNC_RESET</td>
<td>Reset SYNC_REF_FLAG</td>
</tr>
<tr>
<td>%Qr.m.c.9</td>
<td>MODULO_RESET</td>
<td>Reset MODULO_FLAG</td>
</tr>
</tbody>
</table>

The following table presents the **valid_ref** bit in bold which is an element of the %QWr.m.c.0 function enabling word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%QWr.m.c.0.0</td>
<td>VALID_SYNC</td>
<td>Synchronization and start authorization for the counting function via the IN_SYNC input</td>
</tr>
<tr>
<td>%QWr.m.c.0.1</td>
<td>VALID_REF</td>
<td>Operation authorization for the internal preset function</td>
</tr>
<tr>
<td>%QWr.m.c.0.2</td>
<td>VALID_ENABLE</td>
<td>Authorization of the counter enable via the IN_EN input</td>
</tr>
<tr>
<td>%QWr.m.c.0.3</td>
<td>VALID_CAPT_0</td>
<td>Capture authorization in the capture0 register</td>
</tr>
<tr>
<td>%QWr.m.c.0.4</td>
<td>VALID_CAPT_1</td>
<td>Capture authorization in the capture1 register</td>
</tr>
<tr>
<td>%QWr.m.c.0.5</td>
<td>COMPARE_ENABLE</td>
<td>Comparators operation authorization</td>
</tr>
<tr>
<td>%QWr.m.c.0.6</td>
<td>COMPARE_SUSPEND</td>
<td>Comparator frozen at its last value</td>
</tr>
</tbody>
</table>
The following table presents the homing principle:

<table>
<thead>
<tr>
<th>Edge</th>
<th>Status of the valid_ref bit (%QWr.m.c.0.1)</th>
<th>Status of the counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homing condition edge (depending on the configuration)</td>
<td>Set to 0</td>
<td>Not preset</td>
</tr>
<tr>
<td>Homing condition edge (depending on the configuration)</td>
<td>Set to 1</td>
<td>Preset</td>
</tr>
<tr>
<td>Rising edge on force_ref bit (%Qr.m.c.5)</td>
<td>Set to 0 or 1</td>
<td>Preset</td>
</tr>
</tbody>
</table>

When the preset occurs consequently to the preset condition, the application can react using:
- either the SYNC_REF_FLAG input (%IWr.m.c.0.2) (see page 77)
- or the EVT_SYNC_PRESET input (%IWr.m.c.10.2) (see page 79).

Enable Function

This function is used to authorize changes to the current counter value depending on the status of the IN_EN physical input.

This function is used in the following counting modes:
- Pulse width modulation
- Modulo loop counter
- One shot counter
- Free large counter

The following table presents the force_enable bit in bold which is an element of the %Qr.m.c.d output command word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Qr.m.c.0</td>
<td>OUTPUT_0</td>
<td>Forces OUTPUT_0 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.1</td>
<td>OUTPUT_1</td>
<td>Forces OUTPUT_1 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.2</td>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>Implementation of output 0 function block</td>
</tr>
<tr>
<td>%Qr.m.c.3</td>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>Implementation of output 1 function block</td>
</tr>
<tr>
<td>%Qr.m.c.4</td>
<td>FORCE_SYNC</td>
<td>Counting function synchronization and start</td>
</tr>
<tr>
<td>%Qr.m.c.5</td>
<td>FORCE_REF</td>
<td>Set to preset counter value</td>
</tr>
<tr>
<td>%Qr.m.c.6</td>
<td>FORCE_ENABLE</td>
<td>Implementation of counter</td>
</tr>
<tr>
<td>%Qr.m.c.7</td>
<td>FORCE_RESET</td>
<td>Reset counter</td>
</tr>
<tr>
<td>%Qr.m.c.8</td>
<td>SYNC_RESET</td>
<td>Reset SYNC_REF_FLAG</td>
</tr>
<tr>
<td>%Qr.m.c.9</td>
<td>MODULO_RESET</td>
<td>Reset MODULO_FLAG</td>
</tr>
</tbody>
</table>
The following table presents the valid_enable bit in bold which is an element of the %QWr.m.c.0 function enabling word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%QWr.m.c.0.0</td>
<td>VALID_SYNC</td>
<td>Synchronization and start authorization for the counting function via the IN_SYNC input</td>
</tr>
<tr>
<td>%QWr.m.c.0.1</td>
<td>VALID_REF</td>
<td>Operation authorization for the internal preset function</td>
</tr>
<tr>
<td>%QWr.m.c.0.2</td>
<td>VALID_ENABLE</td>
<td>Authorization of the counter enable via the IN_EN input</td>
</tr>
<tr>
<td>%QWr.m.c.0.3</td>
<td>VALID_CAPT_0</td>
<td>Capture authorization in the capture0 register</td>
</tr>
<tr>
<td>%QWr.m.c.0.4</td>
<td>VALID_CAPT_1</td>
<td>Capture authorization in the capture1 register</td>
</tr>
<tr>
<td>%QWr.m.c.0.5</td>
<td>COMPARE_ENABLE</td>
<td>Comparators operation authorization</td>
</tr>
<tr>
<td>%QWr.m.c.0.6</td>
<td>COMPARE_SUSPEND</td>
<td>Comparator frozen at its last value</td>
</tr>
</tbody>
</table>

The following table presents the validation principle:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Status of the valid_enable bit (%QWr.m.c.0.2) and force_enable bit (%Qr.m.c.6)</th>
<th>Status of the counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_EN set to 1</td>
<td>The 2 bits are set to 0</td>
<td>Not counting (frozen)</td>
</tr>
<tr>
<td>IN_EN set to 1</td>
<td>At least one of the two bits is set to 1</td>
<td>Counting (free)</td>
</tr>
</tbody>
</table>
Reset to 0 Function

This function is used to load the value 0 into the counter via software command.

This function is used in the following counting modes:

- Free large counter
- Modulo loop counter
- One shot counter

The following table presents the `force_reset` bit in bold which is an element of the `%Qr.m.c.d` output command word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Qr.m.c.0</td>
<td>OUTPUT_0</td>
<td>Forces OUTPUT_0 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.1</td>
<td>OUTPUT_1</td>
<td>Forces OUTPUT_1 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.2</td>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>Implementation of output 0 function block</td>
</tr>
<tr>
<td>%Qr.m.c.3</td>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>Implementation of output 1 function block</td>
</tr>
<tr>
<td>%Qr.m.c.4</td>
<td>FORCE_SYNC</td>
<td>Counting function synchronization and start</td>
</tr>
<tr>
<td>%Qr.m.c.5</td>
<td>FORCE_REF</td>
<td>Set to preset counter value</td>
</tr>
<tr>
<td>%Qr.m.c.6</td>
<td>FORCE_ENABLE</td>
<td>Implementation of counter</td>
</tr>
<tr>
<td>%Qr.m.c.7</td>
<td>FORCE_RESET</td>
<td>Reset counter</td>
</tr>
<tr>
<td>%Qr.m.c.8</td>
<td>SYNC_RESET</td>
<td>Reset SYNC_REF_FLAG</td>
</tr>
<tr>
<td>%Qr.m.c.9</td>
<td>MODULO_RESET</td>
<td>Reset MODULO_FLAG</td>
</tr>
</tbody>
</table>

The function is only activated by the rising edge of the `force_reset` bit (%Qr.m.c.7). There is no `valid_reset` bit because the function is not activated by any physical input.
Capture Function

This function allows to store the current counter value into a capture register upon an external condition.

Each BMX EHC 0200 module channel has 2 capture registers:

- capture0
- capture1.

The capture function is used in the following counting modes:

- Modulo loop counter
- Free large counter.

In the modulo loop counter mode, only the capture0 function is available.

The function enables to record the current counter value according to the synchronisation condition.

If the IN_SYNC input receives the sensitive edge of synchronization (see page 69), the current counter value is stored into the capt_0_val register (%Idr.m.c.14). The valid_capt_0 bit (%QWr.m.c.0.3) must be set to 1 to operate.

When the synchronization is required at the same time (with the valid_sync bit set to 1) the storage into the capt_0_val register occurs just before resetting the current counter value.

In the free large counter mode, both capture0 and capture1 registers are available.

The capture1 function always stores the current counter value into the capt_1_val register (%Idr.m.c.16) as soon as the IN_CAP input receives a rising edge. The valid_capt_1 bit (%QWr.m.c.0.4) must be set to 1 to operate.

The capture0 function can be configured as one of the following 2 conditions:

- Preset condition
- Falling edge of the IN_CAP input.

The valid_capt_0 bit (%QWr.m.c.0.3) must be set to 1 to operate.

If the capture0 function is configured as the preset condition, the function stores the current counter value into the capt_0_val register (%Idr.m.c4) when the defined preset condition (see page 71) occurs.

When the preset is required at the same time (with the valid_ref bit set to 1) the storage into the capt_0_val register occurs just before loading the current counter value at the preset value.

In all cases, the current counter value must be valid before the capture event (the validity bit (%IWr.m.c.0.3) set to 1)
Modulo Flag and Synchronization Flag

At a Glance

This section presents the operation of the bits relating to the following events:
- Synchronization or counter homing event, depending on the counting mode.
- Counter rollovers the modulo or its limits in forward or reverse.

The table below presents the counting modes that may activate synchronization, homing and modulo events:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Counting mode concerned</th>
</tr>
</thead>
<tbody>
<tr>
<td>sync_ref_flag bit (%IWr.m.c.0.2)</td>
<td>• Free Large counter: When the counter presets</td>
</tr>
<tr>
<td></td>
<td>• Modulo loop counter: When the counter resets</td>
</tr>
<tr>
<td></td>
<td>• One shot counting: When the counter presets and starts</td>
</tr>
<tr>
<td>modulo_flag bit (%IWr.m.c.0.1)</td>
<td>• Modulo loop counter: When the counter rollovers the modulo or 0</td>
</tr>
<tr>
<td></td>
<td>• Free large counter: When the counter rollovers its limits.</td>
</tr>
</tbody>
</table>

Operation of the Flag Bits

The synchronization or homing event’s flag bit is set to 1 when a counter synchronization or homing occurs.

The modulo event's flag bit is set to 1 in the following counting modes:
- Modulo loop counter mode: the flag bit is set to 1 when the counter rollovers the modulo
- Free large counter mode: the flag bit is set to 1 when the counter rollovers its limits in forward or reverse

Location of the Flag Bits

The following table presents the modulo_flag and sync_ref_flag bits which are elements of the %IWr.m.c.d status word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.0</td>
<td>RUN</td>
<td>The counter operates in one shot mode only</td>
</tr>
<tr>
<td>%IWr.m.c.0.1</td>
<td>MODULO_FLAG</td>
<td>Flag set to 1 by a modulo switch event</td>
</tr>
<tr>
<td>%IWr.m.c.0.2</td>
<td>SYNC_REF_FLAG</td>
<td>Flag set to 1 by a preset or synchronization event</td>
</tr>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>The current numerical value is valid</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The current numerical value is locked at the upper threshold value</td>
</tr>
<tr>
<td>%IWr.m.c.0.5</td>
<td>LOW_LIMIT</td>
<td>The current numerical value is locked at the lower threshold value</td>
</tr>
</tbody>
</table>
Resetting the Flag Bits to 0

The user application must reset the flag bit to 0 (if it is active) by using the appropriate command bit from the following two bits:

- sync_reset (%IWr.m.c.8) bit to reset the synchronization or homing event's flag bit to 0
- modulo_reset (%IWr.m.c.9) bit to reset the modulo reached event's flag bit to 0

Location of Reset to 0 Commands

The following table presents the sync_reset and modulo_reset bits which are elements of the %Qr.m.c.d output command word:

<table>
<thead>
<tr>
<th>Language object</th>
<th>Standard symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Qr.m.c.0</td>
<td>OUTPUT_0</td>
<td>Forces OUTPUT_0 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.1</td>
<td>OUTPUT_1</td>
<td>Forces OUTPUT_1 to level 1</td>
</tr>
<tr>
<td>%Qr.m.c.2</td>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>Implementation of output 0 function block</td>
</tr>
<tr>
<td>%Qr.m.c.3</td>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>Implementation of output 1 function block</td>
</tr>
<tr>
<td>%Qr.m.c.4</td>
<td>FORCE_SYNC</td>
<td>Counting function synchronization and start</td>
</tr>
<tr>
<td>%Qr.m.c.5</td>
<td>FORCE_REF</td>
<td>Set to preset counter value</td>
</tr>
<tr>
<td>%Qr.m.c.6</td>
<td>FORCE_ENABLE</td>
<td>Implementation of counter</td>
</tr>
<tr>
<td>%Qr.m.c.7</td>
<td>FORCE_RESET</td>
<td>Reset counter</td>
</tr>
<tr>
<td>%Qr.m.c.8</td>
<td>SYNC_RESET</td>
<td>Reset SYNC_REF_FLAG</td>
</tr>
<tr>
<td>%Qr.m.c.9</td>
<td>MODULO_RESET</td>
<td>Reset MODULO_FLAG</td>
</tr>
</tbody>
</table>
Sending Counting Events to the Application

At a Glance

The event task number must be declared in the module’s configuration screen to enable the events sending.

The BMX EHC 0200 module has eight event sources contained in the `events_source` word at the address `%IWr.m.c.10:

<table>
<thead>
<tr>
<th>Address</th>
<th>Standard Symbol</th>
<th>Description</th>
<th>Counting mode concerned</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.10.0</td>
<td>EVT_RUN</td>
<td>Event due to start of counting.</td>
<td>One Shot Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.1</td>
<td>EVT_MODULO</td>
<td>Event due to counter being equal to modulo value - 1 or equal to value 0.</td>
<td>Modulo Loop Counter Mode, Free Large Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.2</td>
<td>EVT_SYNC_PRESET</td>
<td>Event due to a synchronization or counter homing.</td>
<td>Event Counting mode, One Shot Counter mode, Modulo Loop Counter mode, Free Large Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.3</td>
<td>EVT_COUNTER_LOW</td>
<td>Event due to counter being less than the lower threshold.</td>
<td>Frequency mode, Event Counting mode, Period Measuring mode, Ratio mode, One Shot Counter mode, Modulo Loop Counter mode, Free Large Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.4</td>
<td>EVT_COUNTER_WINDOW</td>
<td>Event due to counter being between the upper and lower thresholds.</td>
<td>Frequency mode, Event Counting mode, Period Measuring mode, Ratio mode, One Shot Counter mode, Modulo Loop Counter mode, Free Large Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.5</td>
<td>EVT_COUNTER_HIGH</td>
<td>Event due to counter being greater than the upper threshold.</td>
<td>Frequency mode, Event Counting mode, Period Measuring mode, Ratio mode, One Shot Counter mode, Modulo Loop Counter mode, Free Large Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.6</td>
<td>EVT_CAPT_0</td>
<td>Event due to capture 0.</td>
<td>Modulo Loop Counter Mode, Free Large Counter mode</td>
</tr>
<tr>
<td>%IWr.m.c.10.7</td>
<td>EVT_CAPT_1</td>
<td>Event due to capture 1.</td>
<td>Free Large Counter mode</td>
</tr>
</tbody>
</table>
All the events sent by the module, whatever their source, call the same single event task in the PLC.

There is normally only one type of event indicated per call.

The `evt_sources` word (%IWr.m.c.10) is updated at the start of the event task processing.

### Enabling Events

In order for a source to produce an event, the validation bit corresponding to the event must be set to 1:

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%QWr.m.c.1.0</td>
<td>Start of counting event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.1</td>
<td>Counter rollovering modulo, 0 or its limits event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.2</td>
<td>Synchronization or counter homing event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.3</td>
<td>Counter less than lower threshold event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.4</td>
<td>Counter between the upper and lower thresholds event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.5</td>
<td>Counter greater than upper threshold event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.6</td>
<td>Capture 0 event validation bit.</td>
</tr>
<tr>
<td>%QWr.m.c.1.7</td>
<td>Capture 1 event validation bit.</td>
</tr>
</tbody>
</table>

### Input Interface

The event only has one input interface. This interface is only updated at the start of the event task processing. The interface consists of:

- The `evt_sources` word (%IWr.m.c.10)
- The current value of the counter during the event (or an approximate value) contained in the `counter_value` word (%Idr.m.c.12)
- The `capt_0_val` register (%Idr.m.c.14) updated if the event is the capture 0
- The `capt_1_val` register (%Idr.m.c.16) updated if the event is the capture 1
Operating Limits

Each counter channel can produce a maximum of one event per millisecond, but this flow may be slowed down by simultaneously sending events to several modules on the PLC bus.

Each counter channel has a four slot transmission buffer which can be used to store several events while waiting to be sent.

If the channel is unable to send all of the internally produced events, the overrun_evt bit (%IWr.m.c.10.8) of the evt_sources word is set to 1.
Section 6.2
BMX EHC 0200 Module Operation Modes

Subject of this Section
This section deals with the different counting modes of the BMX EHC 0200 module.

What is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMX EHC 0200 Module Operation in Frequency Mode</td>
<td>83</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in Event Counting Mode</td>
<td>84</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in Period Measuring Mode</td>
<td>86</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in Ratio Mode</td>
<td>89</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in One Shot Counter Mode</td>
<td>92</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in Modulo Loop Counter Mode</td>
<td>95</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in Free Large Counter Mode</td>
<td>99</td>
</tr>
<tr>
<td>BMX EHC 0200 Module Operation in Pulse Width Modulation Mode</td>
<td>107</td>
</tr>
</tbody>
</table>
BMX EHC 0200 Module Operation in Frequency Mode

At a Glance
Using the frequency mode allows you to measure an event frequency, speed, rate and flow.

Basic Principle
In this mode, the module monitors the pulses applied only to the IN_A input and calculates the number of pulses in time intervals of 1 s. The current frequency is then shown in number of events per second (hertz). The counting register is updated at the end of each 10 ms interval.

Counter Status Bits in Frequency Mode
The table below shows the composition of the counter's %IWr.m.c.0 status word in frequency mode.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value (frequency) and compare status registers contain valid data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the bit is set to 1, the data is valid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the bit is set to 0, the data is not valid.</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The bit is set to 1 when the input frequency signal is out of range.</td>
</tr>
</tbody>
</table>

Type of the IODDT
In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits
The maximum frequency that the module can measure on the IN_A input is 60 kHz. Beyond 60 kHz, the counting register value may decrease until it reaches 0. Beyond 60 kHz and up to the real cut-off frequency of 100 kHz, the module may indicate that it has exceeded the frequency limit.

When there is a variation in frequency, the value restoration time is 1 s with a value precision of 1 Hz. When there is a very significant variation in frequency, an accelerator enables you to restore the frequency value with a precision of 10 Hz in 0.1 s.

The maximum duty cycle at 60 KHz is 60%.

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in Event Counting Mode

At a Glance
Using the event counting mode allows you to determine the number of events received in a scattered manner.

Basic Principle
In this mode, the counter assesses the number of pulses applied at the IN_A input, at time intervals defined by the user. The counting register is updated at the end of each interval with the number of events received.

It is possible to use the IN_SYNC input over a time interval, provided that the validation bit is set to 1. This restarts the event counting for a new predefined time interval. Depending on the selection made by the user, the time interval starts at the rising edge or at the falling edge on the IN_SYNC input.

Operation
The trend diagram below illustrates the counting process in event counting mode:
Counter Status Bits in Event Counting Mode

The table below shows the composition of the counter’s %IWr.m.c.0 status word in event counting mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.2</td>
<td>SYNC_REF_FLAG</td>
<td>The bit is set to 1 when the internal time base has been synchronized. The bit is set to 0 when the sync_reset command is received (rising edge of the %Qr.m.c.8 bit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value (events number) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The bit is set to 1 when the number of received events exceeds the counter size. The bit is reset to 0 at the next period if the limit is not reached.</td>
</tr>
<tr>
<td>%IWr.m.c.0.5</td>
<td>LOW_LIMIT</td>
<td>The bit is set to 1 when more than one synchronization is received within 5 ms period. The bit is reset to 0 at the next period if the limit is not reached.</td>
</tr>
</tbody>
</table>

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The module counts the pulses applied at the IN_A input every time the pulse duration is greater than 5 μs (without debounce filter).

The synchronization of the counter must not be done more than one time per 5 ms.

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in Period Measuring Mode

At a Glance
Using the period measuring mode allows to:
- determine the duration of an event.
- determine the time between two events.
- set and measure the execution time for a process.

Basic Principle
This counting mode consists of two sub-modes:
- Rising edge to falling edge mode (edge to opposite): allows you to measure the duration of an event.
- Rising edge to rising edge mode (edge to edge): allows you to measure the length of time between two events.

The user may also use the IN_SYNC input to enable or stop a measurement. It is also possible to specify a time out value in the configuration screen. This function allows to stop a measurement that exceeds this time out. In this case, the counting register is not valid until the next complete measurement.

The units used to measure the length of time of an event or between two events are defined by the user (1 μs, 100 μs or 1 ms).

Edge to Opposite Mode
In this sub-mode, the measurement is taken between the rising edge and the falling edge of the IN_A input. The counting register is updated as soon as the falling edge is detected.

The trend diagram below shows the operating mode of the edge to opposite sub-mode:
**Edge to Edge Mode**

In this sub-mode, the measurement is taken between two rising edges of the IN_A input. The counting register is updated as soon as the second rising edge is detected.

The trend diagram below shows the operating mode of the edge to edge sub-mode:

![Diagram of Edge to Edge Mode](image)

**Using the Synchronization Function**

The trend diagram below illustrates the period measurement counting process in edge to opposite mode when using the synchronization function:

![Diagram of Synchronization Function](image)

1. The falling edge of the IN_SYNC input stops measurement C.
2. This pulse is not measured because the IN_SYNC input is not at the high level.

**NOTE:** The valid_sync bit (%QWr.m.c.0.0) must be set to 1 to enable the IN_SYNC input. If the IN_SYNC input is not wired, the application must force the setting of the force_sync bit (%Qr.m.c.4) to 1 to authorize the measurements.
Counter Status Bits in Period Measuring Mode

The table below shows the composition of the counter's %IWr.m.c.0 status word in period measuring mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value (period value) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The bit is set to 1 when the measured period exceeds the user-defined timeout. The bit is reset to 0 at the next period if the timeout is not reached.</td>
</tr>
<tr>
<td>%IWr.m.c.0.5</td>
<td>LOW_LIMIT</td>
<td>The bit is set to 1 when more than one measure occurs within 5 ms period. The bit is reset to 0 at the next period if the limit is not reached.</td>
</tr>
</tbody>
</table>

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The module can perform a maximum of 1 measurement every 5 ms.

The shortest pulse that can be measured is 100 μs, even if the unit defined by the user is 1 μs.

The maximum duration that can be measured is 1,073,741,823 units of time (unit defined by the user).

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in Ratio Mode

At a Glance

The ratio mode only uses the IN_A and IN_B inputs. This counting mode consists of two sub-modes:

- Ratio 1: is used to divide two frequencies (Frequency IN_A / Frequency IN_B) and is useful, for example, in applications such as flowmeters and mixers.
- Ratio 2: is used to subtract two frequencies (Frequency IN_A - Frequency IN_B) and is used in the same applications but which require more precise adjustment (closer frequencies).

NOTE: A positive value indicates that the frequency measured on the IN_A input is greater than the frequency measured on the IN_B input. A negative value indicates that the frequency measured on the IN_A input is less than the frequency measured on the IN_B input.

Ratio 1 Mode

The figure below shows BMX EHC 0200 module operation in Ratio 1 mode.

In this mode, the counter evaluates the ratio between the number of rising edges of the IN_A input and the number of rising edges of the IN_B input over a period of 1 s. The register value is updated every 10 ms.

An absolute limit value is declared on the configuration screen. If this limit value is exceeded, the counter_value register (%IDr.m.c.12) is disabled by setting the validity bit (%IWr.m.c.0.3) to 0.

If no frequency is applied to the IN_A or IN_B inputs, the counter_value register (%IDr.m.c.12) is disabled by setting the validity bit (%IWr.m.c.0.3) to 0.

NOTE: The ratio 1 mode presents the results in thousandths in order to have greater level of precision (where 2,000 is displayed, this corresponds to a value of 2).
Ratio 2 Mode

The figure below shows BMX EHC 0200 module operation in Ratio 2 mode.

In this mode, the counter evaluates the difference between the number of rising edges of the IN_A input and the number of rising edges of the IN_B input over a period of 1 s. The counter_value register (%Idr.m.c.12) is updated at the end of each 10 ms interval.

An absolute limit value is declared on the configuration screen. If this limit value is exceeded, the counter_value register (%Idr.m.c.12) is disabled and the validity bit (%IWr.m.c.0.3) to 0.

Counter Status Bits in Ratio Mode

The table below shows bits that are used by the status word %IWr.m.c.0 when the counter is configured in ratio mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value (ratio value) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The bit signals a error when the ratio exceeds the absolute limit. The bit is set to 1 when frequency to IN_A becomes too fast. The bit is reset to 0 when the frequency to IN_A remains correct.</td>
</tr>
<tr>
<td>%IWr.m.c.0.5</td>
<td>LOW_LIMIT</td>
<td>The bit signals a error when the ratio exceeds the absolute limit. The bit is set to 1 when frequency to IN_B becomes too fast. The bit is reset to 0 when the frequency to IN_B remains correct.</td>
</tr>
</tbody>
</table>

Type of the IODDT

In this mode, the type of the IODDT must be T_SIGNED_CPT_BMX.
Operating Limits

The maximum frequency that the module can measure on the IN_A and IN_B inputs is 60 kHz. The measured values are between -60,000,000,000 and +60,000,000,000.

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in One Shot Counter Mode

At a Glance
Using the one shot counter mode allows you to quantify a group of parts.

Basic Principle
In this mode, activating the synchronization function starts the counter which, starting from a value defined by the user in the adjust screen (preset value), decreases with every pulse applied to the IN_A input until it reaches the value 0. Downcounting is made possible when the enable function is activated. The counting register is thus updated every 1 ms.

One basic use of this mode is, using an output, to indicate the end of a group of operations (when the counter reaches 0).

Operation
The trend diagram below illustrates the counting process in one shot counter mode:
In the trend diagram above, we can see that the counter is set to the preset value at the IN_SYNC input’s rising edge. Then, the counter decrements the counting register with every pulse applied to the IN_A input. When the register is set to 0, the counter awaits a new signal from the IN_SYNC input. The IN_A input pulses have no effect on the register value as long as the counter is set to 0.

The enable function must be activated during the counting by:

- setting to 1 the force_enable bit
- setting to 1 the valid_enable bit and when the IN_EN input is at the high level

When the enable function is deactivated, the last value reported in the counting register is maintained and the counter ignores the pulses applied to the IN_A input. However, it does not ignore the IN_SYNC input status.

Each time the counter starts a downcounting operation, the run bit switches to the high level. It switches to the low level when the register value reaches 0.

**NOTE:** The pulses applied to IN_SYNC and IN_EN inputs are only taken into account when the inputs are enabled (see page 73).

The value defined by the user (preset value) is contained in the word %MDr.m.c.6. The user may change this value by specifying the value of this word by configuring the parameter in the adjust screen or by using the `WRITE_PARAM(IODDT_VAR1)` function. IODDT_VAR1 is of the type T_UNSIGNED_CPT_BMX. This value change is only taken into account by the module after one of the following conditions has been established:

- At the next synchronization if the counter is stopped (run bit set to 0)
- At the second synchronization if the counter is activated (run bit set to 1).

### Counter Status Bits in One shot Counter Mode

The table below shows bits that are used by the status word %IWr.m.c.0 when the counter is configured in one shot counter mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.0</td>
<td>RUN</td>
<td>The bit is set to 1 when the counter is running. The bit is set to 0 when the counter is stopped.</td>
</tr>
<tr>
<td>%IWr.m.c.0.2</td>
<td>SYNC_REF_FLAG</td>
<td>The bit is set to 1 when the counter has been set to the preset value and (re)started. The bit is reset to 0 when the sync_reset command is received (rising edge of the %Qr.m.c.8 bit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.</td>
</tr>
</tbody>
</table>

### Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.
Operating Limits

The maximum frequency that can be applied to the IN_SYNC input is 1 pulse every 5 ms.
The maximum value defined by the user (preset value) is 4,294,967,295.

**NOTE:** You have to check the validity bit (IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in Modulo Loop Counter Mode

At a Glance

The use of the modulo loop counter mode is recommended for packaging and labeling applications for which actions are repeated for series of moving objects.

Basic Principle

In the upcounting direction, the counter increases until it reaches the modulo value -1, the modulo value being defined by the user. At the following pulse in the counting direction, the counter is reset to 0 and the counting resumes.

In the downcounting direction, the counter decreases until it reaches 0. At the next pulse in the counting direction, the counter is reset to the the modulo value -1, the modulo value being defined by the user. The downcounting may then be resumed.

The enable function must be activated during the counting by:

- Setting to 1 the `force_enable` bit (%Qr.m.c.6)
- Setting to 1 the `valid_enable` bit (%QWr.m.c.0.2) when the IN_EN input is at the high level

When the enable function is deactivated, the last value reported in the counting register is maintained and the counter ignores the pulses applied to the IN_A input. However, it does not ignore the preset condition.

In the modulo loop counter mode, the counter must be synchronized at least one time to operate. The current counter value is cleared each time the synchronization occurs.

The current counter value can be recorded into the capture0 register (see page 76) when the condition of synchronization occurs (see page 69).

The modulo value defined by the user is contained in the `modulo_value` word %MDr.m.c.4. The user may change this value by specifying the value of this word:

- In the adjust screen
- In the application, using the `WRITE_PARAM(IODDT_VAR1)` Function. IODDT_VAR1 is of the type `T_UNSIGNED_CPT_BMX`.

The new modulo value is acknowledged if one of the two following conditions is met:

- The synchronization is activated
- The counter rollovers the value 0 in the downcounting direction or the modulo value -1 (this value is the modulo value recorded before editing the new modulo value) in the upcounting direction.
Counting Interface

In this mode, the user may select one of the following counting configurations:

- A = Up, B = Down (default configuration)
- A = Impulse, B = Direction
- Normal Quadrature X1
- Normal Quadrature X2
- Normal Quadrature X4
- Reverse Quadrature X1
- Reverse Quadrature X2
- Reverse Quadrature X4.

The following table shows the upcounting and downcounting principle according to the selected configuration:

<table>
<thead>
<tr>
<th>Selected configuration</th>
<th>Upcounting condition</th>
<th>Downcounting condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = Up, B = Down</td>
<td>Rising edge at the IN_A input.</td>
<td>Rising edge at the IN_B input.</td>
</tr>
<tr>
<td>A = Impulse, B = Direction</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
<td>Rising edge at the IN_A input and high state at the IN_B input.</td>
</tr>
<tr>
<td>Normal Quadrature X1</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td>Normal Quadrature X2</td>
<td>Rising edge at the IN_A input and high state at the IN_B input.</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td>Normal Quadrature X4</td>
<td>Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input.</td>
<td>Rising edge at the IN_A input and high state at the IN_B input.</td>
</tr>
<tr>
<td>Reserve Quadrature X1</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td>Reserve Quadrature X2</td>
<td>Falling edge at the IN_A input and low state at the IN_B input. Rising edge at the IN_A input and high state at the IN_B input.</td>
<td>Rising edge at the IN_A input and high state at the IN_B input.</td>
</tr>
</tbody>
</table>
Selected configuration | Upcounting condition | Downcounting condition
--- | --- | ---
Reserve Quadrature X4 | Falling edge at the IN_A input and low state at the IN_B input. Low state at the IN_A input and rising edge at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input. High state at the IN_A input and falling edge at the IN_B input. | Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input. Low state at the IN_A input and falling edge at the IN_B input. |

**Operation**

The trend dithe modulo counting process in the configuration by default (IN_A = counting, IN_B = downcounting):
Counter Status Bits in Modulo Loop Counter Mode

The table below shows the composition of the counter’s %IWr.m.c.0 status word in modulo loop counter mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.1</td>
<td>MODULO_FLAG</td>
<td>The bit is set to 1 when the counter rollovers the modulo and is.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The bit is reset to 0 when the command MODULO_RESET(%Qr.m.c.9) is received.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(rising edge of the MODULO_RESET bit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.2</td>
<td>SYNC_REF_FLAG</td>
<td>The bit is set to 1 when the counter have been set to 0 and (re)started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The bit is reset to 0 when the command SYNC_RESET(%Qr.m.c.8) is received.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(rising edge of the SYNC_RESET bit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value and compare</td>
</tr>
<tr>
<td></td>
<td></td>
<td>status registers contain valid data. If the bit is set to 1, the data is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>valid. If the bit is set to 0, the data is not valid.</td>
</tr>
</tbody>
</table>

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The maximum frequency that can be applied to the IN_SYNC input is 1 pulse every 5 ms.

The maximum frequency for the modulo event is once every 5 ms.

The maximum value for the defined modulo value and the counter is 4,294,967,295.

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in Free Large Counter Mode

At a Glance

The use of the free large counter mode is especially recommended for axis monitoring or labeling where the incoming position of each part has to be learned.

Basic Principle

The upcounting (or downcounting) starts as soon as the homing function is completed. The enable function must be activated during the counting by:

- Setting to 1 the force_enable bit (%Qr.m.c.6)
- Setting to 1 the valid_enable bit (%Qwr.m.c.0.2) when the IN_EN input is at the high level.

When the enable function is deactivated, the last value reported in the counting register is maintained and the counter ignores the pulses applied to the IN_A input. However, it does not ignore the preset condition.

In the free large counter mode, the counter must be preset at least one time to operate. The current counter value is loaded with the preset_value each time the preset condition occurs.

The current counter can be recorded into the capture0 register when the preset condition occurs or using the IN_CAP input.

The current counter value can be stored into the capture1 register using the IN_CAP input.

For further information, you may see the synchronization function (see page 69) and the capture function (see page 76).

In the free large counter mode, the counting register is updated at 1 ms intervals.
Counting Configurations

In this mode, the user may select one of the following counting configurations:

- A = Up, B = Down (default configuration)
- A = Impulse, B = Direction
- Normal Quadrature X1
- Normal Quadrature X2
- Normal Quadrature X4
- Reverse Quadrature X1
- Reverse Quadrature X2
- Reverse Quadrature X4

The following table shows the upcounting and downcounting principle according to the selected configuration:

<table>
<thead>
<tr>
<th>Selected configuration</th>
<th>Upcounting condition</th>
<th>Downcounting condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = Up, B = Down</td>
<td>Rising edge at the IN_A input.</td>
<td>Rising edge at the IN_B input.</td>
</tr>
<tr>
<td>A = Impulse, B = Direction</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
<td>Rising edge at the IN_A input and high state at the IN_B input.</td>
</tr>
<tr>
<td>Normal Quadrature X1</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td>Normal Quadrature X2</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td>Normal Quadrature X4</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
<td>Rising edge at the IN_A input and high level at the IN_B input.</td>
</tr>
<tr>
<td>Reserve Quadrature X1</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td>Reserve Quadrature X2</td>
<td>Falling edge at the IN_A input and low state at the IN_B input.</td>
<td>Rising edge at the IN_A input and low state at the IN_B input.</td>
</tr>
<tr>
<td></td>
<td>Rising edge at the IN_A input and high level at the IN_B input.</td>
<td>Rising edge at the IN_A input and high state at the IN_B input.</td>
</tr>
<tr>
<td></td>
<td>Raising edge at the IN_A input and rising edge at the IN_B input.</td>
<td>High state at the IN_A input and falling edge at the IN_B input.</td>
</tr>
<tr>
<td></td>
<td>Low state at the IN_A input and high state at the IN_B input.</td>
<td>Low state at the IN_A input and falling edge at the IN_B input.</td>
</tr>
<tr>
<td></td>
<td>Raising edge at the IN_A input and low state at the IN_B input.</td>
<td>Raising edge at the IN_A input and high level at the IN_B input.</td>
</tr>
<tr>
<td></td>
<td>High state at the IN_A input and falling edge at the IN_B input.</td>
<td>Low state at the IN_A input and falling edge at the IN_B input.</td>
</tr>
</tbody>
</table>
Homing Function

This function allows to record the `current_counter_value` register in the `capt_0_val` register and/or to set the `current_counter_value` register to the user-predefined parameter `preset_value`.

The value defined by the user as `preset_value` is contained in the `%MDr.m.c.4` word.

The user may change this value by specifying the value of this word:
- In the adjust screen
- In the application, by using the `WRITE_PARAM(IODDT_VAR1)` function. `IODDT_VAR1` is of the type `T_SIGNED_CPT_BMX`.

For further information, you may see the homing function (see page 71) and the capture function (see page 76).

The module configuration enables you to select the following homing conditions:
- Rising edge of the IN_SYNC input (default)
- Rising edge of the IN_REF input
- Rising edge of the IN_SYNC input at the IN_REF input’s high level

<table>
<thead>
<tr>
<th>Selected configuration</th>
<th>Upcounting condition</th>
<th>Downcounting condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserve Quadrature X4</td>
<td>Failing edge at the IN_A input and low state at the IN_B input. Low state at the IN_A input and rising edge at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input. High state at the IN_A input and falling edge at the IN_B input.</td>
<td>Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input. Low state at the IN_A input and falling edge at the IN_B input.</td>
</tr>
</tbody>
</table>

For further information, you may see the homing function (see page 71) and the capture function (see page 76).
- First rising edge of the IN_SYNC input and high level at the IN_REF input

- First rising edge of the IN_SYNC input and low level at the IN_REF input
**Operation**

The trend diagram below illustrates the counting process for a free large counter in the configuration by default:

- **IN_A input (pulses)**
- **IN_B input (pulses)**
- **Preset condition**
- **Enable condition**
- **IN_CAP input**
- **Counter value**

The diagram shows the progression of pulses, preset conditions, enable conditions, and the counter value over time. Points X1, X2, and X3 mark significant transitions or states in the counting process.
Behavior at the Counting Limits

When the upper or lower limit is exceeded, the counter behaves differently according to its configuration.

In the lock on limits default configuration, the counting register maintains the limit value once it has been reached and the counting validity bit changes to 0 until the next preset condition occurs:

NOTE: Overflow or underflow is indicated by two bits LOW_LIMIT and HIGH_LIMIT until the application reloads the counting value predefined by the user (force_ref bit set to 1 or preset condition true). The upcounting or downcounting may therefore resume.

In the rollover configuration, the counting register switches to the opposite limit value when one of the two limits is exceeded:
**Slack Delete**

In the free large counter mode, the counter may apply a hysteresis if the rotation is inverted. The `hysteresis` parameter configured with the adjust screen defines the number of points that are not acknowledged by the counter during the rotation inversion. This aims to take into account the slack between the encoder/motor axis and the mechanical axis (e.g., an encoder measuring the position of a mat).

This behavior is described in the following figure:

The value defined by the user as the `Hysteresis (slack)` value is contained in the `%MWr.m.c.9` word. The user may change this value by specifying the value of this word (this value is from 0 to 255):

- In the adjust screen
- In the application by using the `WRITE_PARAM(IODDT_VAR1)` Function. `IODDT_VAR1` is of the type `T_SIGNED_CPT_BMX`. 
Counter Status Bits in Free Large Counter Mode

The table below shows the composition of the counter's %IWr.m.c.0 status word in free large counter mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.1</td>
<td>MODULO_FLAG</td>
<td>The bit status changes in the rollover mode. The bit is set to 1 when the counter rollovers its limits (-2,147,483,648 or +2,147,483,647). The bit is reset to 0 when the command MODULO_RESET (%Qr.m.c.9) is received (rising edge of the MODULO_RESET bit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.2</td>
<td>SYNC_REF_FLAG</td>
<td>The bit is set to 1 when the counter have been set to the preset value and (re)started. The bit is reset to 0 when the command SYNC_RESET (%Qr.m.c.8) is received (rising edge of the SYNC_RESET bit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the counter current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The bit status changes in the lock on limits mode. The bit is set to 1 when the counter reaches +2,147,483,647. The bit is reset to 0 when the counter presets or resets.</td>
</tr>
<tr>
<td>%IWr.m.c.0.5</td>
<td>LOW_LIMIT</td>
<td>The bit status changes in the lock on limits mode. The bit is set to 1 when the counter reaches -2,147,483,648. The bit is reset to 0 when the counter presets or resets.</td>
</tr>
</tbody>
</table>

Type of the IODDT

In this mode, the type of the IODDT must be T_SIGNED_CPT_BMX.

Operating Limits

The shortest pulse applied to the IN_SYNC input is 100 μs.
The maximum homing event frequency is once every 5 ms.
The counter value is between -2,147,483,648 and +2,147,483,647.

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.
BMX EHC 0200 Module Operation in Pulse Width Modulation Mode

At a Glance

In this operating mode, the module uses an internal clock generator to supply a periodic signal at the module's Q0 output. Only the Q0 output is concerned by this mode as the Q1 output is independent of this mode.

Basic Principle

The `output_block_0_enable` command bit (Qr.m.c.2) must be set to 1 in order to enable a modulation at the Q0 output.

The active validation function enables the operation of the internal clock generator that produces the output signal to be validated.

The active synchronization function enables the output signal to be synchronized by resetting to 0 the internal clock generator.

The wave form of the output signal depends on:

- The `pwm_frequency` value (QDr.m.c.6): it defines the frequency from 0.1 Hz (value is equal to 1) to 4 KHz (value is equal to 40,000), in increments of 0.1 Hz
- The `pwm_duty` value (QWr.m.c.8): it defines the duty cycle from 5 % (value is equal to 1) to 95 % (value is equal to 19) in increments of 5 %.

The following figure shows the operation of the module in the pulse width modulation mode:
Counter Status Bits in Pulse Width Modulation Mode

The table below shows the composition of the counter’s %IWr.m.c.0 status word in pulse width modulation mode:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%IWr.m.c.0.3</td>
<td>VALIDITY</td>
<td>Validity bit is used to indicate that the output data (frequency and duty cycle) under current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.</td>
</tr>
<tr>
<td>%IWr.m.c.0.4</td>
<td>HIGH_LIMIT</td>
<td>The output frequency or the duty cycle is out of range (high limit).</td>
</tr>
<tr>
<td>%IWr.m.c.0.5</td>
<td>LOW_LIMIT</td>
<td>The output frequency or the duty cycle is out of range (low limit).</td>
</tr>
</tbody>
</table>

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The maximum output frequency is 4 kHz.

The maximum frequency that can be applied to the IN_SYNC input is 1 pulse every 5 ms. The Q0 driver is "source type", therefore a load resistance is required to switch the output signal Q0 to 0 V using the correct frequency. We recommend a load resistance of 250 Ω. The allowed duty cycle varies according to the frequency of the Q0 output.

The table below shows duty cycle values according to the selected frequency. These values must be observed for normal operation:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Duty cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1... 250 Hz</td>
<td>95% - 5%</td>
</tr>
<tr>
<td>251... 500 Hz</td>
<td>90% - 10%</td>
</tr>
<tr>
<td>501... 1 000 Hz</td>
<td>80% - 20%</td>
</tr>
<tr>
<td>1001... 1 500 Hz</td>
<td>70% - 30%</td>
</tr>
<tr>
<td>1501... 2 000 Hz</td>
<td>60% - 40%</td>
</tr>
<tr>
<td>2 001... 2 500 Hz</td>
<td>50%</td>
</tr>
<tr>
<td>2 5001... 4 000 Hz</td>
<td>50% (See following note)</td>
</tr>
</tbody>
</table>

NOTE: If the frequency and the duty cycle do not respect this table, the output and the validity bit (%IWr.m.c.0.3) remains in the low state.

NOTE: You have to check the validity bit (%IWr.m.c.0.3) before taking into account the numerical values such as the counter and the capture registers. Only the validity bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

NOTE: From 2501 Hz to 4000 Hz, the 50% ratio is not guaranteed on output.
Part IV
Counting Module BMX EHC 0200 Software Implementation

Subject of this Part
This part describes the software implementation and functions of the BMX EHC 0200 counting module.

NOTE: This part concerns also the hardened module BMX EHC 0200H.

What Is in This Part?
This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Software Implementation Methodology for BMX EHC xxxx Counting Modules</td>
<td>111</td>
</tr>
<tr>
<td>8</td>
<td>Accessing the Functional Screens of the BMX EHC xxxx Counting Modules</td>
<td>113</td>
</tr>
<tr>
<td>9</td>
<td>Configuration of the BMX EHC 0200 Counting Modules</td>
<td>119</td>
</tr>
<tr>
<td>10</td>
<td>BMX EHC xxxx Counting Module Settings</td>
<td>143</td>
</tr>
<tr>
<td>11</td>
<td>Debugging the BMX EHC 0200 Counting Modules</td>
<td>151</td>
</tr>
<tr>
<td>12</td>
<td>Display of BMX EHC xxxx Counting Module Error</td>
<td>167</td>
</tr>
<tr>
<td>13</td>
<td>The Language Objects of the Counting Function</td>
<td>173</td>
</tr>
</tbody>
</table>
Chapter 7
Software Implementation Methodology for BMX EHC xxxx Counting Modules

Installation Methodology

At a Glance

The software installation of the BMX EHC **** counting modules is carried out from the various Control Expert editors:

- in offline mode,
- in online mode.

The following order of installation phases is recommended but it is possible to change the order of certain phases (for example, starting with the configuration phase).

Installation Phases

The following table shows the different installation phases:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration of variables</td>
<td>Declaration of IODDT-type variables for the application-specific modules and variables of the project.</td>
<td>Offline(1)</td>
</tr>
<tr>
<td>Programming</td>
<td>Project programming.</td>
<td>Offline(1)</td>
</tr>
<tr>
<td>Configuration</td>
<td>Declaration of modules.</td>
<td>Offline</td>
</tr>
<tr>
<td></td>
<td>Module channel configuration</td>
<td>Offline(1)</td>
</tr>
<tr>
<td></td>
<td>Entering the configuration parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: All the parameters are configurable online except the event parameter.</td>
<td></td>
</tr>
<tr>
<td>Association</td>
<td>Association of IODDTs with the channels configured (variable editor)</td>
<td>Offline(1)</td>
</tr>
<tr>
<td>Build</td>
<td>Project generation (analysis and editing of links)</td>
<td>Offline</td>
</tr>
<tr>
<td>Transfer</td>
<td>Transfer project to PLC</td>
<td>Online</td>
</tr>
<tr>
<td>Adjustment/Debugging</td>
<td>Debug project from debug screens, animation tables</td>
<td>Online</td>
</tr>
<tr>
<td></td>
<td>Debugging the program and adjustment parameters</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Building documentation file and printing miscellaneous information relating to the project</td>
<td>Online(1)</td>
</tr>
<tr>
<td>Phase</td>
<td>Description</td>
<td>Mode</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Operation/Diagnostic</td>
<td>Displaying miscellaneous information necessary for supervisory control of the project</td>
<td>Online</td>
</tr>
<tr>
<td></td>
<td>Diagnostics of project and modules</td>
<td></td>
</tr>
</tbody>
</table>

Key:

(1) These various phases can also be performed in online mode
Chapter 8
Accessing the Functional Screens of the BMX EHC xxxx Counting Modules

Subject of this Chapter
This chapter describes the various functional screens of the BMX EHC xxxx counting modules to which the user has access.

What Is in This Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing the Functional Screens of the BMX EHC 0200 Counting Modules</td>
<td>114</td>
</tr>
<tr>
<td>Description of the Counting Module Screens</td>
<td>116</td>
</tr>
</tbody>
</table>
Accessing the Functional Screens of the BMX EHC 0200 Counting Modules

At a Glance

This section describes how to access the functional screens of the BMX EHC 0200 counting modules.

Procedure

To access the screens, execute the following actions:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Expand the Configuration directory in the project browser.  
  
  **Result:** the following screen appears: |
| 2    | Double-click on the PLC Bus directory.  
  
  **Result:** the following screen appears: |
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Double-click on the counting module. <strong>Result</strong>: the module screen appears:</td>
</tr>
</tbody>
</table>

![Module Screen](image)

<table>
<thead>
<tr>
<th>Label</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

3-channel generic counter Version 1.00
Description of the Counting Module Screens

Introduction
The various available screens for the BMX EHC 0200 counting modules are:

- Configuration screen
- Adjust screen
- Debug screen (can only be accessed in online mode)
- Faults screen (can only be accessed in online mode)

Description of the Screens
The following diagram presents the counting modules configuration screen.
The following table presents the parts of the various screens.

<table>
<thead>
<tr>
<th>Number</th>
<th>Element</th>
<th>Function</th>
</tr>
</thead>
</table>
| 1      | Tabs               | The tab in the foreground indicates the mode in progress (Configuration in this example). Every mode can be selected using the respective tab. The available modes are:  
- **Configuration**  
- **Adjust**  
- **Debug** (which can only be accessed in online mode)  
- **Faults** (which can only be accessed in online mode) |
| 2      | Heading area       | Provides an abbreviation as a reminder of the module and module status in online mode (LEDs). |
| 3      | Module area        | Is used:  
- By clicking on the reference number, to display the tabs:  
  - **Description** which gives the characteristics of the device.  
  - **I/O Objects** or **Device DDT** depending on the I/O data type selected at module insertion in the Control Expert project. |
|        | Channel area       | Is used:  
- By clicking on the channel (Counter) number, to display the tabs:  
  - **Configuration** which gives the characteristics of the channel. By default in topological I/O data model, no function is configured. By default in device DDT data model, all channels are **Frequency Mode** configured and a channel can not be set to **None**.  
  - **Adjust**: consists of various sections to be completed (parameter values), displayed according to the choice of counting function.  
  - **Debug**: displays the status of the inputs and outputs, as well as the various parameters of the current counting function (in online mode).  
  - **Fault**: which shows the device errors (in online mode). |
| 4      | General parameters area | Allows you to select the counting function and the task associated with the channel:  
- **Function**: counting function among those available for the modules involved. Depending on this choice, the headings of the configuration area may differ.  
- **Task**: defines the task through which the channel's implicit exchange objects will be exchanged.  
These choices are only possible in offline mode. |
| 5      | Parameters in progress area | This area has various functionalities which depend upon the current mode:  
- **Configuration**: is used to configure the channel parameters.  
- **Adjust**: consists of various sections to be completed (parameter values), displayed according to the choice of counting function.  
- **Debug**: displays the status of the inputs and outputs, as well as the various parameters of the current counting function.  
- **Fault**: displays the errors that have occurred on the counting channels. |
Chapter 9
Configuration of the BMX EHC 0200 Counting Modules

Subject of this Chapter
This chapter deals with the configuration of the BMX EHC 0200 counting modules. This configuration can be accessed from the Configuration tab on the functional screens of BMX EHC 0200 (see page 116) modules.

What Is in This Chapter?
This chapter contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Configuration Screen for BMX EHC xxxx Counting Modules</td>
<td>120</td>
</tr>
<tr>
<td>9.2</td>
<td>Configuration of Modes for the BMX EHC 0200 Module</td>
<td>123</td>
</tr>
</tbody>
</table>
Section 9.1
Configuration Screen for BMX EHC xxxx Counting Modules

Configuration Screen for BMX EHC 0200 Counting Modules in a Modicon M340 Local Rack

At a Glance
This section presents the configuration screen for BMX EHC 0200 counting modules.
Illustration

The figure below presents the configuration screen for the BMX EHC 0200 module in modulo loop counter mode:

NOTE: When adding a BMX EHC 0200 in a local rack the default function is **Frequency mode**
**Description of the Screen**

The following table presents the various parts of the above screen:

<table>
<thead>
<tr>
<th>Number</th>
<th>Element</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tab</td>
<td>The tab in the foreground indicates the current mode. The current mode is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>therefore the configuration mode in this example.</td>
</tr>
<tr>
<td>2</td>
<td>Label field</td>
<td>This field contains the name of each variable that may be configured. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td>field may not be modified.</td>
</tr>
<tr>
<td>3</td>
<td>Symbol field</td>
<td>This field contains the address of the variable in the application. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td>field may not be modified.</td>
</tr>
<tr>
<td>4</td>
<td>Value field</td>
<td>If this field has a downward pointing arrow, you can select the value of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>each variable from various possible values in this field. The various</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values can be accessed by clicking on the arrow. A drop-down menu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>containing all the possible values is displayed and the user may then select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the required value of the variable.</td>
</tr>
<tr>
<td>5</td>
<td>Unit field</td>
<td>This field contains the unit of each variable that may be configured. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td>field may not be modified.</td>
</tr>
</tbody>
</table>
Section 9.2
Configuration of Modes for the BMX EHC 0200 Module

Subject of this Section
This section deals with the configuration of the modes of the BMX EHC 0200 counting modules.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Mode Configuration</td>
<td>124</td>
</tr>
<tr>
<td>Event Counting Mode Configuration</td>
<td>126</td>
</tr>
<tr>
<td>Period Measuring Mode Configuration</td>
<td>128</td>
</tr>
<tr>
<td>Ratio Mode Configuration</td>
<td>130</td>
</tr>
<tr>
<td>One Shot Counter Mode Configuration</td>
<td>132</td>
</tr>
<tr>
<td>Modulo Loop Counter Mode Configuration</td>
<td>134</td>
</tr>
<tr>
<td>Free Large Counter Mode Configuration</td>
<td>137</td>
</tr>
<tr>
<td>Pulse Width Modulation Mode Configuration</td>
<td>140</td>
</tr>
</tbody>
</table>
Frequency Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r, m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the frequency mode configurable elements.

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2</td>
<td>Frequency mode. The value of the least significant byte of this word is 1.</td>
</tr>
<tr>
<td>IN_A input filter</td>
<td>%KWr.m.c.3</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: high.</td>
</tr>
<tr>
<td>Input power supply fault</td>
<td>%KWr.m.c.2.8</td>
<td>General input/output fault (bit set to 0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Local (bit set to 1)</td>
</tr>
<tr>
<td>Scale factor</td>
<td>%KWr.m.c.6</td>
<td>Edit (value in the range 1...255)</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td></td>
</tr>
<tr>
<td>Output block 0</td>
<td>%KWr.m.c.17</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Label</td>
<td>Address in the configuration</td>
<td>Configurable values</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output block 1</td>
<td>%Kwr.m.c.19</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%Kwr.m.c.21.1</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%Kwr.m.c.21.2</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%Kwr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%Kwr.m.c.21.3</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback 1</td>
<td>%Kwr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 0</td>
<td>%Kwr.m.c.21.5</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%Kwr.m.c.21.6</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply fault</td>
<td>%Kwr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Offline (bit set to 1)</td>
</tr>
<tr>
<td>Pulse width 0</td>
<td>%Kwr.m.c.18</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Pulse width 1</td>
<td>%Kwr.m.c.20</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Event</td>
<td>%Kwr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>
Configuration of the Counting Modules

Event Counting Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r, m, and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:
- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the event counting mode configurable elements

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2</td>
<td>Event counting mode. The value of the least significant byte of this word is 2.</td>
</tr>
<tr>
<td>IN_A input filter</td>
<td>%KWr.m.c.3</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>IN_SYNC input filter</td>
<td>%KWr.m.c.4</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>Input power supply</td>
<td>%KWr.m.c.2.8</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td>fault</td>
<td></td>
<td>Local (bit set to 1)</td>
</tr>
<tr>
<td>Synchronization edge</td>
<td>%KWr.m.c.10.8</td>
<td>Rising edge at IN_SYNC (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Falling edge at IN_SYNC (bit set to 1)</td>
</tr>
<tr>
<td>Time base</td>
<td>%KWr.m.c.7</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0: 0.1 s,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: 1 s,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: 10 s,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: 1 min</td>
</tr>
</tbody>
</table>
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output block 0</td>
<td>%KWr.m.c.17</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Output block 1</td>
<td>%KWr.m.c.19</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%KWr.m.c.21.1</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%KWr.m.c.21.2</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%KWr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%KWr.m.c.21.3</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback 1</td>
<td>%KWr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 0</td>
<td>%KWr.m.c.21.5</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%KWr.m.c.21.6</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply fault</td>
<td>%KWr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Offline (bit set to 1)</td>
</tr>
<tr>
<td>Pulse width 0</td>
<td>%KWr.m.c.18</td>
<td>Edit (value in the range 1..65535)</td>
</tr>
<tr>
<td>Pulse width 1</td>
<td>%KWr.m.c.20</td>
<td>Edit (value in the range 1..65535)</td>
</tr>
<tr>
<td>Event</td>
<td>%KWr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)</td>
</tr>
<tr>
<td>Event number</td>
<td>%KWr.m.c.0</td>
<td>Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>
Configuration of the Counting Modules

Period Measuring Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW). The parameters r, m, and c shown in the following tables represent the topologic addressing of the module. Each parameter has the following significance:
- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the period measuring mode configurable elements.

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2 (least significant byte)</td>
<td>Period measuring mode. The value of the least significant byte of this word is 3.</td>
</tr>
</tbody>
</table>
| IN_A input filter      | %KWr.m.c.3 (least significant byte) | The least significant byte can take the following values:  
- 0: none,  
- 1: low,  
- 2: medium,  
- 3: high. |
| IN_SYNC input filter   | %KWr.m.c.4 (least significant byte) | The least significant byte can take the following values:  
- 0: none,  
- 1: low,  
- 2: medium,  
- 3: high. |
| Input power supply fault | %KWr.m.c.2.8 | General input/output fault (bit set to 0)  
Local (bit set to 1) |
| Resolution             | %KWr.m.c.8 (most significant byte) | The most significant byte can take the following values:  
- 0: 1 µs,  
- 1: 100 µs,  
- 2: 1 ms. |
| Mode                   | %KWr.m.c.8 (least significant byte) | The least significant byte can take the following values:  
- 0: From one edge to the same edge at input IN_A,  
- 1: From one edge to the opposite edge at input IN_A. |
| Time out               | %KDr.m.c.14 | 0...1 073 741 823 |
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
</table>
| Output block 0 | %KWr.m.c.17 | This word can take the following values:  
- 0: off,  
- 1: low counter,  
- 2: counter in a window,  
- 3: High counter,  
- 4: pulse = less than the lower threshold (LT),  
- 5: pulse = greater than the lower threshold (LT),  
- 6: pulse = less than the upper threshold (UT),  
- 7: pulse = greater than the upper threshold (UT). |
| Output block 1 | %KWr.m.c.19 | This word can take the following values:  
- 0: off,  
- 1: low counter,  
- 2: counter in a window,  
- 3: High counter,  
- 4: pulse = less than the lower threshold (LT),  
- 5: pulse = greater than the lower threshold (LT),  
- 6: pulse = less than the upper threshold (UT),  
- 7: pulse = greater than the upper threshold (UT). |
| Polarity 0 | %KWr.m.c.21.1 | Polarity + (bit set to 0)  
Polarity - (bit set to 1) |
| Polarity 1 | %KWr.m.c.21.2 | Polarity + (bit set to 0)  
Polarity - (bit set to 1) |
| Fault recovery | %KWr.m.c.21.0 | Automatic reaction (bit set to 1)  
Activated (bit set to 0) |
| Fallback 0 | %KWr.m.c.21.3 | None (bit set to 0)  
With (bit set to 1) |
| Fallback 1 | %KWr.m.c.21.4 | None (bit set to 0)  
With (bit set to 1) |
| Fallback value 0 | %KWr.m.c.21.5 | 0 (bit set to 0)  
1 (bit set to 1) |
| Fallback value 1 | %KWr.m.c.21.6 | 0 (bit set to 0)  
1 (bit set to 1) |
| Output power supply fault | %KWr.m.c.21.9 | General input/output fault (bit set to 0)  
Offline (bit set to 1) |
| Pulse width 0 | %KWr.m.c.18 | Edit (value in the range 1...65535) |
| Pulse width 1 | %KWr.m.c.20 | Edit (value in the range 1...65535) |
| Event | %KWr.m.c.0 | Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)  
Deactivated (all bits of the most significant byte of this word are set to 1) |
**Configuration of the Counting Modules**

**Ratio Mode Configuration**

**At a Glance**

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r, m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

**Configuration Objects**

The table below presents ratio mode configurable elements.

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2</td>
<td>The least significant byte of this word can take the following values in this mode:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 4: ratio 1 mode,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 5: ratio 2 mode.</td>
</tr>
<tr>
<td>IN_A input filter</td>
<td>%KWr.m.c.3</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>IN_B input filter</td>
<td>%KWr.m.c.3</td>
<td>The most significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(most significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>Input power supply fault</td>
<td>%KWr.m.c.2.8</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Local (bit set to 1)</td>
</tr>
<tr>
<td>Scale factor</td>
<td>%KWr.m.c.6</td>
<td>Edit (value in the range 1...255)</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td></td>
</tr>
<tr>
<td>Absolute limit</td>
<td>%KDr.m.c.12</td>
<td>Edit</td>
</tr>
</tbody>
</table>
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output block 0</td>
<td>%KWr.m.c.17</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Output block 1</td>
<td>%KWr.m.c.19</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%KWr.m.c.21.1</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%KWr.m.c.21.2</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%KWr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%KWr.m.c.21.3</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback 1</td>
<td>%KWr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 0</td>
<td>%KWr.m.c.21.5</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%KWr.m.c.21.6</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply</td>
<td>%KWr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td>fault</td>
<td></td>
<td>Offline (bit set to 1)</td>
</tr>
<tr>
<td>Pulse width 0</td>
<td>%KWr.m.c.18</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Pulse width 1</td>
<td>%KWr.m.c.20</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Event</td>
<td>%KWr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)</td>
</tr>
<tr>
<td>Event number</td>
<td></td>
<td>Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>

---

Add any necessary explanations or notes here if needed.
One Shot Counter Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r, m, and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the one shot counter mode configurable elements

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2 (least significant byte)</td>
<td>One shot counter mode. The value of the least significant byte of this word is 6.</td>
</tr>
<tr>
<td>IN_A input filter</td>
<td>%KWr.m.c.3 (least significant byte)</td>
<td>The least significant byte can take the following values: 0: none, 1: low, 2: medium, 3: high.</td>
</tr>
<tr>
<td>IN_SYNC input filter</td>
<td>%KWr.m.c.4 (least significant byte)</td>
<td>The least significant byte can take the following values: 0: none, 1: low, 2: medium, 3: high.</td>
</tr>
<tr>
<td>IN_EN input filter</td>
<td>%KWr.m.c.4 (most significant byte)</td>
<td>The most significant byte can take the following values: 0: none, 1: low, 2: medium, 3: high.</td>
</tr>
<tr>
<td>Input power supply fault</td>
<td>%KWr.m.c.2.8</td>
<td>General input/output fault (bit set to 0) Local (bit set to 1)</td>
</tr>
<tr>
<td>Scale factor</td>
<td>%KWr.m.c.6 (least significant byte)</td>
<td>Edit (value in the range 1...255)</td>
</tr>
<tr>
<td>Synchronization edge</td>
<td>%KWr.m.c.10.8</td>
<td>Rising edge (bit set to 0) Falling edge (bit set to 1)</td>
</tr>
<tr>
<td>Label</td>
<td>Address in the configuration</td>
<td>Configurable values</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output block 0</td>
<td>%KWr.m.c.17</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Output block 1</td>
<td>%KWr.m.c.19</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%KWr.m.c.21.1</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%KWr.m.c.21.2</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%KWr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%KWr.m.c.21.3</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback 1</td>
<td>%KWr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 0</td>
<td>%KWr.m.c.21.5</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%KWr.m.c.21.6</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply</td>
<td>%KWr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td>fault</td>
<td></td>
<td>Offline (bit set to 1)</td>
</tr>
<tr>
<td>Pulse width 0</td>
<td>%KWr.m.c.18</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Pulse width 1</td>
<td>%KWr.m.c.20</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Event</td>
<td>%KWr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)</td>
</tr>
<tr>
<td>Event number</td>
<td></td>
<td>Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>
Modulo Loop Counter Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r, m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents modulo loop counter mode configurable elements.

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2</td>
<td>Modulo loop counter mode. The value of the least significant byte of this word is 7.</td>
</tr>
</tbody>
</table>
| IN_A input filter   | %KWr.m.c.3                   | The least significant byte can take the following values:
|                     | (least significant byte)     | ● 0: none,                                                                            |
|                     |                              | ● 1: low,                                                                             |
|                     |                              | ● 2: medium                                                                          |
|                     |                              | ● 3: high                                                                            |
| IN_A input filter   | %KWr.m.c.3                   | The most significant byte can take the following values:
|                     | (most significant byte)      | ● 0: none,                                                                            |
|                     |                              | ● 1: low,                                                                             |
|                     |                              | ● 2: medium                                                                          |
|                     |                              | ● 3: high                                                                            |
| IN_SYNC input filter| %KWr.m.c.4                   | The least significant byte can take the following values:
|                     | (least significant byte)     | ● 0: none,                                                                            |
|                     |                              | ● 1: low,                                                                             |
|                     |                              | ● 2: medium                                                                          |
|                     |                              | ● 3: high                                                                            |
| IN_EN input filter  | %KWr.m.c.4                   | The most significant byte can take the following values:
|                     | (most significant byte)      | ● 0: none,                                                                            |
|                     |                              | ● 1: low,                                                                             |
|                     |                              | ● 2: medium                                                                          |
|                     |                              | ● 3: high                                                                            |
| Input power supply  | %KWr.m.c.2.8                 | General input/output fault (bit set to 0)                                            |
| fault               |                              | Local (bit set to 1)                                                                  |
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input mode</td>
<td>%Wr.m.c.9</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: A = High, B = Low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: A = Pulse, B = Direction,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: normal quadrature 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: normal quadrature 2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: normal quadrature 4,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: inverse quadrature 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: inverse quadrature 2,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: inverse quadrature 4.</td>
</tr>
<tr>
<td>Scale factor</td>
<td>%Wr.m.c.6</td>
<td>Edit (value in the range 1...255)</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td></td>
</tr>
<tr>
<td>Synchronization edge</td>
<td>%Wr.m.c.10</td>
<td>Rising edge (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td>(most significant byte)</td>
<td>Falling edge (bit set to 1)</td>
</tr>
<tr>
<td>Output block 0</td>
<td>%Wr.m.c.17</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Output block 1</td>
<td>%Wr.m.c.19</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%Wr.m.c.21.1</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%Wr.m.c.21.2</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%Wr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%Wr.m.c.21.3</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback 1</td>
<td>%Wr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
</tbody>
</table>
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fallback value 0</td>
<td>%KWr.m.c.21.5</td>
<td>0 (bit set to 0)&lt;br&gt;1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%KWr.m.c.21.6</td>
<td>0 (bit set to 0)&lt;br&gt;1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply fault</td>
<td>%KWr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)&lt;br&gt;Offline (bit set to 1)</td>
</tr>
<tr>
<td>Pulse width 0</td>
<td>%KWr.m.c.18</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Pulse width 1</td>
<td>%KWr.m.c.20</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Event</td>
<td>%KWr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)&lt;br&gt;Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>
Free Large Counter Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r, m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the free large counter mode configurable elements.

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2</td>
<td>Free large counter mode. The value of the least significant byte of this word is 8.</td>
</tr>
<tr>
<td>IN_A input filter</td>
<td>%KWr.m.c.3</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>IN_B input filter</td>
<td>%KWr.m.c.3</td>
<td>The most significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(most significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>IN_SYNC input filter</td>
<td>%KWr.m.c.4</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>IN_EN input filter</td>
<td>%KWr.m.c.4</td>
<td>The most significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(most significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
<tr>
<td>IN_REF input filter</td>
<td>%KWr.m.c.5</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>- 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: high.</td>
</tr>
</tbody>
</table>
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_CAP input filter</td>
<td>%KWr.m.c.5</td>
<td>The most significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(most significant byte)</td>
<td>• 0: none,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: medium,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: high.</td>
</tr>
<tr>
<td>Input power supply fault</td>
<td>%KWr.m.c.2.8</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Local (bit set to 1)</td>
</tr>
<tr>
<td>Input mode</td>
<td>%KWr.m.c.9</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: A = High, B = Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: A = Pulse, B = Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: normal quadrature 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: normal quadrature 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: normal quadrature 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: inverse quadrature 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: inverse quadrature 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: inverse quadrature 4</td>
</tr>
<tr>
<td>Scale factor</td>
<td>%KWr.m.c.6</td>
<td>Edit (value in the range 1...255)</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td></td>
</tr>
<tr>
<td>Preset mode</td>
<td>%KWr.m.c.10</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td>(least significant byte)</td>
<td>• 0: rising edge at IN_SYNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: rising edge at IN_REF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: rising edge at IN_SYNC and IN_REF at 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: first rising edge at IN_SYNC and IN_REF at 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: first rising edge at IN_SYNC and IN_REF at 0</td>
</tr>
<tr>
<td>Capture 0 settings</td>
<td>%KWr.m.c.16.1</td>
<td>Preset condition (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Falling edge at the IN_CAP input (bit set to 1)</td>
</tr>
<tr>
<td>Output block 0</td>
<td>%KWr.m.c.17</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: low counter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: High counter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 7: pulse = greater than the upper threshold (UT).</td>
</tr>
</tbody>
</table>
### Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output block 1</td>
<td>%KWr.m.c.19</td>
<td>This word can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 0: off,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 1: low counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 2: counter in a window,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 3: High counter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 4: pulse = less than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 5: pulse = greater than the lower threshold (LT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 6: pulse = less than the upper threshold (UT),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● 7: pulse = greater than the upper threshold (UT).</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%KWr.m.c.21.1</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%KWr.m.c.21.2</td>
<td>Polarity + (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%KWr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%KWr.m.c.21.3</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback 1</td>
<td>%KWr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 0</td>
<td>%KWr.m.c.21.5</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%KWr.m.c.21.6</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply</td>
<td>%KWr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td>fault</td>
<td></td>
<td>Offline (bit set to 1)</td>
</tr>
<tr>
<td>Pulse width 0</td>
<td>%KWr.m.c.18</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Pulse width 1</td>
<td>%KWr.m.c.20</td>
<td>Edit (value in the range 1...65535)</td>
</tr>
<tr>
<td>Event</td>
<td>%KWr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)</td>
</tr>
<tr>
<td>Event number</td>
<td></td>
<td>Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>
Pulse Width Modulation Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m: represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the pulse width modulation mode configurable elements.

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting mode</td>
<td>%KWr.m.c.2 (least significant byte)</td>
<td>Pulse width modulation mode. The value of the least significant byte of this word is 9.</td>
</tr>
<tr>
<td>IN_SYNC input filter</td>
<td>%KWr.m.c.4 (least significant byte)</td>
<td>The least significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: none, 1: low, 2: medium, 3: high.</td>
</tr>
<tr>
<td>Synchronization edge</td>
<td>%KWr.m.c.10.8</td>
<td>Rising edge at IN_SYNC (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Falling edge at IN_SYNC (bit set to 1)</td>
</tr>
<tr>
<td>IN_EN input filter</td>
<td>%KWr.m.c.4 (most significant byte)</td>
<td>The most significant byte can take the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: none, 1: low, 2: medium, 3: high.</td>
</tr>
<tr>
<td>Input power supply fault</td>
<td>%KWr.m.c.2.8</td>
<td>General input/output fault (bit set to 0) Local (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 0</td>
<td>%KWr.m.c.21.1</td>
<td>Polarity + (bit set to 0) Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Polarity 1</td>
<td>%KWr.m.c.21.2</td>
<td>Polarity + (bit set to 0) Polarity - (bit set to 1)</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>%KWr.m.c.21.0</td>
<td>Automatic reaction (bit set to 1) Activated (bit set to 0)</td>
</tr>
<tr>
<td>Fallback 0</td>
<td>%KWr.m.c.21.3</td>
<td>None (bit set to 0) With (bit set to 1)</td>
</tr>
</tbody>
</table>
# Configuration of the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Address in the configuration</th>
<th>Configurable values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fallback 1</td>
<td>%KWr.m.c.21.4</td>
<td>None (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 0</td>
<td>%KWr.m.c.21.5</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Fallback value 1</td>
<td>%KWr.m.c.21.6</td>
<td>0 (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (bit set to 1)</td>
</tr>
<tr>
<td>Output power supply fault</td>
<td>%KWr.m.c.2.9</td>
<td>General input/output fault (bit set to 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Offline (bit set to 1)</td>
</tr>
<tr>
<td>Event Event number</td>
<td>%KWr.m.c.0</td>
<td>Activated (if activated is selected, the entered event number is coded on the most significant byte of this word)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deactivated (all bits of the most significant byte of this word are set to 1)</td>
</tr>
</tbody>
</table>
Configuration of the Counting Modules
Chapter 10
BMX EHC xxxx Counting Module Settings

Subject of this Chapter
This chapter deals with the possible settings for the counting modes of the BMX EHC •••• modules. These settings can be accessed from the Configuration tab on the functional screens of BMX EHC •••• modules (see page 116).

What is in This Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjust Screen for BMX EHC 0200 Counting Modules</td>
<td>144</td>
</tr>
<tr>
<td>Setting the Preset Value</td>
<td>146</td>
</tr>
<tr>
<td>Setting the Calibration Factor</td>
<td>147</td>
</tr>
<tr>
<td>Modulo Adjust</td>
<td>148</td>
</tr>
<tr>
<td>Setting the Hysteresis Value</td>
<td>149</td>
</tr>
</tbody>
</table>
Adjust Screen for BMX EHC 0200 Counting Modules

At a Glance
This section presents the adjust screen for BMX EHC 0200 counting modules.

Illustration
The figure below presents the adjust screen for the BMX EHC 0200 module in modulo loop counter mode:
Description of the Screen

The following table presents the various parts of the above screen:

<table>
<thead>
<tr>
<th>Number</th>
<th>Element</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Label field</td>
<td>This field contains the name of each variable that may be adjusted. This field may not be modified and can be accessed in both local and online modes.</td>
</tr>
<tr>
<td>2</td>
<td>Tab</td>
<td>The tab in the foreground indicates the current mode. The current mode is therefore the adjust mode in this example.</td>
</tr>
<tr>
<td>3</td>
<td>Symbol field</td>
<td>This field contains the mnemonics of the variable. This field may not be modified and can be accessed in both offline and online modes.</td>
</tr>
<tr>
<td>4</td>
<td>Initial value field</td>
<td>This field displays the value of the variable that the user has adjusted in offline mode. This field is only accessible in online mode.</td>
</tr>
</tbody>
</table>
| 5      | Value field | The function of this field depends on the mode in which the user is working:  
  • **In offline mode**: this field is used to adjust the variable.  
  • **In online mode**: this field is used to display the current value of the variable. |
| 6      | Unit field  | This field contains the unit of each variable that may be configured. This field may not be modified and can be accessed in both offline and online modes. |
Setting the Preset Value

Introduction
The preset value concerns the following counting modes:

- for the BMX EHC 0200 module:
  - one shot counter mode
  - free large counter mode

Description
The following table shows the preset value setting:

<table>
<thead>
<tr>
<th>Number</th>
<th>Address in the configuration</th>
<th>Value</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preset value</td>
<td>%MDr.m.c.12 (Low)</td>
<td>Edit</td>
<td>0</td>
</tr>
</tbody>
</table>
Setting the Calibration Factor

Introduction

The calibration factor concerns the frequency and ratio modes for the BMX EHC 0200.

Description

The following table shows the calibration factor setting:

<table>
<thead>
<tr>
<th>Number</th>
<th>Address in the configuration</th>
<th>Value</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calibration factor</td>
<td>%MWr.m.c.14</td>
<td>Edit</td>
<td>0</td>
</tr>
</tbody>
</table>
Modulo Adjust

Introduction

The modulo concerns the modulo loop counter modes for the counting modules BMX EHC ****.

Description

The following table shows the modulo adjust:

<table>
<thead>
<tr>
<th>Number</th>
<th>Address in the configuration</th>
<th>Value</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulo</td>
<td>%MDx.y.v.10 (Low)</td>
<td>Edit</td>
<td>0xFFFF</td>
</tr>
</tbody>
</table>
Setting the Hysteresis Value

Introduction

The hysteresis value concerns free large counter mode for module BMX EHC 0200.

Description

The following table shows the setting for the hysteresis value:

<table>
<thead>
<tr>
<th>Number</th>
<th>Address in the configuration</th>
<th>Value</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hysteresis (release value)</td>
<td>%MWr.m.c.9</td>
<td>Edit</td>
<td>0</td>
</tr>
</tbody>
</table>
Subject of this Chapter

This chapter deals with the debugging settings applicable to BMX EHC 0200 modules. These settings can be accessed from the Debug tab on the functional screens of the BMX EHC 0200 (see page 114) modules.

What is in This Chapter?

This chapter contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>Debug Screen for BMX EHC xxxx Counting Modules</td>
<td>152</td>
</tr>
<tr>
<td>11.2</td>
<td>BMX EHC 0200 Module Debugging</td>
<td>155</td>
</tr>
</tbody>
</table>
Section 11.1
Debug Screen for BMX EHC xxxx Counting Modules

At a Glance
This section presents the debug screen for BMX EHC xxxx counting modules. A module's debug screen can only be accessed in online mode.
Illustration

The figure below presents the debug screen for the BMX EHC 0200 module in modulo loop counter mode:
Debugging the Counting Modules

Description of the Screen

The following table presents the various parts of the above screen:

<table>
<thead>
<tr>
<th>Number</th>
<th>Element</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reference</td>
<td>This field contains the address of the variable in the application. This field may not be modified.</td>
</tr>
<tr>
<td>2</td>
<td>Label</td>
<td>This field contains the name of each variable that may be configured. This field may not be modified.</td>
</tr>
<tr>
<td>3</td>
<td>Tab</td>
<td>The tab in the foreground indicates the current mode. The current mode is therefore the debug mode in this example.</td>
</tr>
<tr>
<td>4</td>
<td>Symbol</td>
<td>This field contains the mnemonics of the variable. This field may not be modified.</td>
</tr>
<tr>
<td>5</td>
<td>Value</td>
<td>If the field has a downward pointing arrow, you can select the value of each variable from various possible values in this field. The various values can be accessed by clicking on the arrow. A drop-down menu containing all the possible values is displayed and the user may then select the required value of the variable. If there is no downward pointing arrow, this field simply displays the current value of the variable.</td>
</tr>
</tbody>
</table>
Section 11.2
BMX EHC 0200 Module Debugging

Subject of this Section
This section deals with the debugging of the BMX EHC 0200 counting module modes.

What is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Mode Debugging</td>
<td>156</td>
</tr>
<tr>
<td>Event Counting Mode Debugging</td>
<td>157</td>
</tr>
<tr>
<td>Period Measuring Mode Debugging</td>
<td>158</td>
</tr>
<tr>
<td>Ratio Mode Debugging</td>
<td>159</td>
</tr>
<tr>
<td>One Shot Counter Mode Debugging</td>
<td>160</td>
</tr>
<tr>
<td>Modulo Loop Counter Mode Debugging</td>
<td>161</td>
</tr>
<tr>
<td>Free Large Counter Mode Debugging</td>
<td>163</td>
</tr>
<tr>
<td>Pulse Width Modulation Mode Debugging</td>
<td>165</td>
</tr>
</tbody>
</table>
# Frequency Mode Debugging

## At a Glance

The table below presents the frequency mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency value</td>
<td>%IDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Frequency valid</td>
<td>%IWr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Frequency low</td>
<td>%IWr.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Frequency in window</td>
<td>%IWr.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Frequency high</td>
<td>%IWr.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Frequency in high limit</td>
<td>%IWr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>%Ir.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT (see page 185).
### Event Counting Mode Debugging

**At a Glance**

The table below presents the event counting mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter value</td>
<td>%IDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Counter valid</td>
<td>%IWr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter low</td>
<td>%IWr.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in window</td>
<td>%IWr.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter high</td>
<td>%IWr.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in low limit</td>
<td>%IWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in high limit</td>
<td>%IWr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>%Ir.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input SYNC state</td>
<td>%Ir.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC enable</td>
<td>%QWr.m.c.0.0</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC force</td>
<td>%Qr.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC state</td>
<td>%IWr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC reset</td>
<td>%Qr.m.c.8</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT (see page 185).
## Debugging the Counting Modules

### Period Measuring Mode Debugging

#### At a Glance

The table below presents the period measuring mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period value</td>
<td>%IDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Period valid</td>
<td>%IWr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Period low</td>
<td>%IWr.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Period in window</td>
<td>%IWr.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Period high</td>
<td>%IWr.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Period in low limit</td>
<td>%IWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Period in high limit</td>
<td>%IWr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>%Ir.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input SYNC state</td>
<td>%Ir.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC enable</td>
<td>%QWr.m.c.0.0</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC force</td>
<td>%Qr.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC state</td>
<td>%IWr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC reset</td>
<td>%Qr.m.c.8</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT (see page 185).
## Ratio Mode Debugging

### At a Glance

The table below presents the ratio mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio value</td>
<td>%IDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Ratio valid</td>
<td>%IWr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Ratio low</td>
<td>%IWr.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Ratio in window</td>
<td>%IWr.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Ratio high</td>
<td>%IWr.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Ratio in low limit</td>
<td>%IWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Ratio in high limit</td>
<td>%IWr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>%Ir.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input B state</td>
<td>%Ir.m.c.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_SIGNED_CPT_BMX IODDT (see page 185).
# One Shot Counter Mode Debugging

## At a Glance

The table below presents the one shot counter mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter value</td>
<td>%IDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Counter valid</td>
<td>%IWr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter low</td>
<td>%IWr.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in window</td>
<td>%IWr.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter high</td>
<td>%IWr.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>RUN</td>
<td>%IWr.m.c.0.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>%Ir.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input SYNC state</td>
<td>%Ir.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC enable</td>
<td>%QWr.m.c.0.0</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC force</td>
<td>%Qr.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC state</td>
<td>%IWr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC reset</td>
<td>%Qr.m.c.8</td>
<td>Binary</td>
</tr>
<tr>
<td>Input EN</td>
<td>%Ir.m.c.7</td>
<td>Binary</td>
</tr>
<tr>
<td>EN enable</td>
<td>%QWr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter enable</td>
<td>%Qr.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT (see page 185).
Modulo Loop Counter Mode Debugging

At a Glance

The table below presents the modulo loop counter mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter value</td>
<td>%IDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Counter valid</td>
<td>%IW.r.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter low</td>
<td>%IW.r.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in window</td>
<td>%IW.r.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter high</td>
<td>%IW.r.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in low limit</td>
<td>%IW.r.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in high limit</td>
<td>%IW.r.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture value</td>
<td>%IDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Capture low</td>
<td>%IW.r.m.c.1.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture in window</td>
<td>%IW.r.m.c.1.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture high</td>
<td>%IW.r.m.c.1.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture enable</td>
<td>%QWR.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>%Ir.r.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input B state</td>
<td>%Ir.r.m.c.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Input SYNC state</td>
<td>%Ir.r.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC enable</td>
<td>%QWr.m.c.0.0</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC force</td>
<td>%Qr.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC state</td>
<td>%IW.r.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC reset</td>
<td>%QWr.m.c.8</td>
<td>Binary</td>
</tr>
<tr>
<td>Input EN</td>
<td>%Ir.r.m.c.7</td>
<td>Binary</td>
</tr>
<tr>
<td>EN enable</td>
<td>%QWr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter enable</td>
<td>%Qr.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.r.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.r.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter reset</td>
<td>%Qr.m.c.7</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.r.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.r.m.c.3</td>
<td>Binary</td>
</tr>
</tbody>
</table>
## Debugging the Counting Modules

For a description of each language object refer to `T_UNSIGNED_CPT_BMX IODDT (see page 185).`

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output latch 01enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Modulo state</td>
<td>%IWr.m.c.0.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Modulo reset</td>
<td>%Qr.m.c.9</td>
<td>Binary</td>
</tr>
</tbody>
</table>
Free Large Counter Mode Debugging

At a Glance

The table below presents the free large counter mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter value</td>
<td>$%$Dr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>Counter valid</td>
<td>$%$Wr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter low</td>
<td>$%$Wr.m.c.1.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in window</td>
<td>$%$Wr.m.c.1.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter high</td>
<td>$%$Wr.m.c.1.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in low limit</td>
<td>$%$Wr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter in high limit</td>
<td>$%$Wr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 0 value</td>
<td>$%$Dr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Capture 0 low</td>
<td>$%$Wr.m.c.1.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 0 in window</td>
<td>$%$Wr.m.c.1.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 0 high</td>
<td>$%$Wr.m.c.1.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 0 enable</td>
<td>$%$Qr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 1 value</td>
<td>$%$Dr.m.c.16</td>
<td>Digital</td>
</tr>
<tr>
<td>Capture 1 low</td>
<td>$%$Wr.m.c.1.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 1 in window</td>
<td>$%$Wr.m.c.1.7</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 1 high</td>
<td>$%$Wr.m.c.1.8</td>
<td>Binary</td>
</tr>
<tr>
<td>Capture 1 enable</td>
<td>$%$Qr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input A state</td>
<td>$%$Ir.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input B state</td>
<td>$%$Ir.m.c.5</td>
<td>Binary</td>
</tr>
<tr>
<td>IN_SYNC input</td>
<td>$%$Ir.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Modulo state</td>
<td>$%$Wr.m.c.0.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Modulo reset</td>
<td>$%$Qr.m.c.9</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC state</td>
<td>$%$Wr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC reset</td>
<td>$%$Qr.m.c.8</td>
<td>Binary</td>
</tr>
<tr>
<td>Input EN</td>
<td>$%$Ir.m.c.7</td>
<td>Binary</td>
</tr>
<tr>
<td>EN enable</td>
<td>$%$Wr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter enable</td>
<td>$%$Qr.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Input REF</td>
<td>$%$Ir.m.c.8</td>
<td>Binary</td>
</tr>
<tr>
<td>REF enable</td>
<td>$%$Wr.m.c.0.1</td>
<td>Binary</td>
</tr>
<tr>
<td>REF force</td>
<td>$%$Qr.m.c.5</td>
<td>Binary</td>
</tr>
</tbody>
</table>
### Debugging the Counting Modules

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input CAP</td>
<td>%Ir.m.c.9</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter reset</td>
<td>%Qr.m.c.7</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 state</td>
<td>%Ir.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 state</td>
<td>%Ir.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 1 enable</td>
<td>%Qr.m.c.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Low threshold value</td>
<td>%QDr.m.c.2</td>
<td>Digital</td>
</tr>
<tr>
<td>High threshold value</td>
<td>%QDr.m.c.4</td>
<td>Digital</td>
</tr>
<tr>
<td>Compare enable</td>
<td>%QWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Compare suspend</td>
<td>%QWr.m.c.0.6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_SIGNED_CPT_BMX IODDT (*see page 185*).
Pulse Width Modulation Mode Debugging

At a Glance

The table below presents the pulse width modulation mode debugging elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>Language object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency valid</td>
<td>%IWr.m.c.0.3</td>
<td>Binary</td>
</tr>
<tr>
<td>Frequency in low limit</td>
<td>%IWr.m.c.0.5</td>
<td>Binary</td>
</tr>
<tr>
<td>Frequency in high limit</td>
<td>%IWr.m.c.0.4</td>
<td>Binary</td>
</tr>
<tr>
<td>PWM frequency</td>
<td>%QDr.m.c.6</td>
<td>Digital</td>
</tr>
<tr>
<td>PWM duty</td>
<td>%QWr.m.c.8</td>
<td>Digital</td>
</tr>
<tr>
<td>Input SYNC state</td>
<td>%Ir.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC enable</td>
<td>%QWr.m.c.0.0</td>
<td>Binary</td>
</tr>
<tr>
<td>SYNC force</td>
<td>%Qr.m.c.4</td>
<td>Binary</td>
</tr>
<tr>
<td>Input EN</td>
<td>%Ir.m.c.7</td>
<td>Binary</td>
</tr>
<tr>
<td>EN enable</td>
<td>%QWr.m.c.0.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Counter enable</td>
<td>%Qr.m.c.6</td>
<td>Binary</td>
</tr>
<tr>
<td>Output latch 0 enable</td>
<td>%Qr.m.c.2</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 state</td>
<td>%Ir.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 0 cmd</td>
<td>%Qr.m.c.0</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 state</td>
<td>%Ir.m.c.1</td>
<td>Binary</td>
</tr>
<tr>
<td>Output 1 cmd</td>
<td>%Qr.m.c.1</td>
<td>Binary</td>
</tr>
</tbody>
</table>

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT (see page 185).
Chapter 12
Display of BMX EHC xxxx Counting Module Error

Subject of this Chapter
This chapter deals with the display of possible errors for the BMX EHC•••• modules.

What is in This Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Display Screen for BMX EHC 0200 Counting Modules</td>
<td>168</td>
</tr>
<tr>
<td>Faults Diagnostics Display</td>
<td>170</td>
</tr>
<tr>
<td>List of Error</td>
<td>171</td>
</tr>
</tbody>
</table>
Fault Display Screen for BMX EHC 0200 Counting Modules

At a Glance

This section presents the fault display screen for BMX EHC 0200 counting modules. A module’s fault display screen may only be accessed in online mode.

Illustration

The figure below presents the fault display screen for the BMX EHC 0200 module in modulo loop counter mode.
Description of the Screen

The following table presents the various parts of the above screen.

<table>
<thead>
<tr>
<th>Number</th>
<th>Element</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Internal faults field</td>
<td>This field displays the module’s active internal faults.</td>
</tr>
<tr>
<td>2</td>
<td>Tab</td>
<td>The tab in the foreground indicates the current mode. The current mode is therefore the fault display mode in this example.</td>
</tr>
<tr>
<td>3</td>
<td>External faults field</td>
<td>This field displays the module’s active external faults.</td>
</tr>
<tr>
<td>4</td>
<td>Other faults field</td>
<td>This field displays the module’s active faults, other than internal and external faults.</td>
</tr>
</tbody>
</table>
Faults Diagnostics Display

At a Glance

The diagnostic screens (see page 113) on the module or channel are only accessible in connected mode. When an un-masked fault appears, it is reported:

- in the configuration screen of the rack, with the presence of a red square in the position of the faulty counting module,
- in all screens at module level (Description and Fault tabs),
  - in the module field with the LED
- in all channel level screens (Configuration, Adjustment, Debug and Fault tabs),
  - in the module zone with the LED
  - in the channel zone with the fault LED
- in the fault screen that is accessed by the Fault where the fault diagnostics are described.

The fault is also signaled:

- On the module, on the central display,
- by dedicated language objects: CH_ERROR (%Ir.m.c.ERR) and MOD_ERROR (%Ir.m.MOD.ERR), %MWr.m.MOD.2, etc. and status words.

NOTE: Even if the fault is masked, it is reported by the flashing of the I/O LED and in the fault screen.
List of Error

At a Glance

The messages displayed on the diagnostics screens are used to assist with debugging. These messages must be concise and are sometimes ambiguous (as different faults may have the same consequences).

These diagnostics are on two levels: module and channel, the latter being the most explicit.

The lists below show the message headings with suggestions for identifying issues.

List of the Module Error Messages

The table below provides a list of the module error messages.

<table>
<thead>
<tr>
<th>Fault indicated</th>
<th>Possible interpretation and/or action.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module failure</td>
<td>The module has a error. Check the module mounting. Change the module.</td>
</tr>
<tr>
<td>Faulty channel(s)</td>
<td>One or more channels have a fault. Refer to channel diagnostics.</td>
</tr>
<tr>
<td>Self-test</td>
<td>The module is running a self-test. Wait until the self-test is complete.</td>
</tr>
<tr>
<td>Different hardware and software configurations</td>
<td>There is a lack of compatibility between the module configured and the module in the rack. Make the hardware configuration and the software configuration compatible.</td>
</tr>
<tr>
<td>Module is missing or off</td>
<td>Install the module. Fasten the mounting screws.</td>
</tr>
</tbody>
</table>

BMX EHC 0200 Module Error

The table below provides a list of error that may appear on the BMX EHC 0200 module.

<table>
<thead>
<tr>
<th>Language object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%MWr.m.c.2.0</td>
<td>External fault at inputs</td>
</tr>
<tr>
<td>%MWr.m.c.2.1</td>
<td>External fault at outputs</td>
</tr>
<tr>
<td>%MWr.m.c.2.4</td>
<td>Internal error or self-testing.</td>
</tr>
<tr>
<td>%MWr.m.c.2.5</td>
<td>Configuration Fault</td>
</tr>
<tr>
<td>%MWr.m.c.2.6</td>
<td>Communication Error</td>
</tr>
<tr>
<td>%MWr.m.c.2.7</td>
<td>Application fault</td>
</tr>
<tr>
<td>%MWr.m.c.3.2</td>
<td>Sensor power supply fault</td>
</tr>
<tr>
<td>%MWr.m.c.3.3</td>
<td>Actuator supply fault</td>
</tr>
<tr>
<td>%MWr.m.c.3.4</td>
<td>Short circuit on output 0</td>
</tr>
<tr>
<td>%MWr.m.c.3.5</td>
<td>Short circuit on output 1</td>
</tr>
</tbody>
</table>
### List of Channel Error Messages

The table below gives the list of error messages at channel level.

<table>
<thead>
<tr>
<th>Fault indicated. Other consequences.</th>
<th>Possible interpretation and/or action.</th>
</tr>
</thead>
<tbody>
<tr>
<td>External fault or counting input fault:</td>
<td>Check the sensor connections.</td>
</tr>
<tr>
<td>- encoder or proximity sensor supply fault</td>
<td>Check the sensor connections.</td>
</tr>
<tr>
<td>- line break or short circuit of at least one encoder differential signal (1A, 1B, 1Z)</td>
<td>Check the sensor power supply.</td>
</tr>
<tr>
<td>- specific fault on absolute encoder</td>
<td>Check the sensor operation.</td>
</tr>
<tr>
<td>Outputs are set to 0 in automatic mode.</td>
<td>Delete the fault and acknowledge if the fault storing is configured.</td>
</tr>
<tr>
<td><em>Invalid measurement</em> message.</td>
<td>Counting pulses or incremental encoder: preset or reset to acknowledge the <em>Invalid measurement</em> message.</td>
</tr>
<tr>
<td>Counting application fault:</td>
<td>Diagnose the fault more precisely (external causes).</td>
</tr>
<tr>
<td>- measurement overrun</td>
<td>Check the application again, if necessary.</td>
</tr>
<tr>
<td>- overspeed</td>
<td>Delete the fault and acknowledge if the fault storing is configured.</td>
</tr>
<tr>
<td>Outputs are set to 0 in automatic mode.</td>
<td>Counting pulses or incremental encoder: preset or reset to 0 to acknowledge the <em>Invalid measurement</em> message.</td>
</tr>
<tr>
<td><em>Invalid measurement</em> message.</td>
<td></td>
</tr>
<tr>
<td>Auxiliary input/output fault:</td>
<td>Check the output connections</td>
</tr>
<tr>
<td>- power supply</td>
<td>Check the input/output power supply (24V)</td>
</tr>
<tr>
<td>- short circuit of at least one output</td>
<td>Diagnose the fault more precisely (external causes)</td>
</tr>
<tr>
<td>Outputs are set to 0 in automatic mode.</td>
<td>Delete the fault and acknowledge if the fault storing is configured</td>
</tr>
<tr>
<td>Internal error or channel self-testing:</td>
<td>Module fault has gone down to channel level. Refer to module level diagnostics.</td>
</tr>
<tr>
<td>- module faulty</td>
<td></td>
</tr>
<tr>
<td>- module missing or off</td>
<td></td>
</tr>
<tr>
<td>- module running self-test</td>
<td></td>
</tr>
<tr>
<td>Different hardware and software configurations</td>
<td>Module fault has gone down to channel level. Refer to module level diagnostics.</td>
</tr>
<tr>
<td>Invalid software configuration:</td>
<td>Check and modify the configuration constants.</td>
</tr>
<tr>
<td>- incorrect constant</td>
<td></td>
</tr>
<tr>
<td>- bit combination not associated with any configuration</td>
<td></td>
</tr>
<tr>
<td>Communication error</td>
<td>Check the connections between the racks.</td>
</tr>
<tr>
<td>Application fault: refusal to configure or adjust</td>
<td>Diagnose the fault more precisely.</td>
</tr>
</tbody>
</table>
Chapter 13
The Language Objects of the Counting Function

Subject of this Chapter
This chapter describes the language objects associated to the counting tasks as well as the different ways of using them.

What Is in This Chapter?
This chapter contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>The Language Objects and IODDT of the Counting Function</td>
<td>174</td>
</tr>
<tr>
<td>13.2</td>
<td>Language Objects and IODDT Associated with the Counting Function of the BMX EHC xxxx Modules.</td>
<td>184</td>
</tr>
<tr>
<td>13.3</td>
<td>The IODDT Type T_GEN_MOD Applicable to All Modules</td>
<td>192</td>
</tr>
<tr>
<td>13.4</td>
<td>Device DDTs Associated with the Counting Function of the BMX EHC xxxx Modules.</td>
<td>194</td>
</tr>
</tbody>
</table>
Section 13.1
The Language Objects and IODDT of the Counting Function

Subject of this Section
This section describes the general features of the language objects and IODDT of the counting function.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing Language Objects for Application-Specific Counting</td>
<td>175</td>
</tr>
<tr>
<td>Implicit Exchange Language Objects Associated with the Application-Specific Function</td>
<td>176</td>
</tr>
<tr>
<td>Explicit Exchange Language Objects Associated with the Application-Specific Function</td>
<td>177</td>
</tr>
<tr>
<td>Management of Exchanges and Reports with Explicit Objects</td>
<td>179</td>
</tr>
</tbody>
</table>
Introducing Language Objects for Application-Specific Counting

General

The counting modules have only two associated IODDTs. These IODDTs are predefined by the manufacturer and contains language objects for inputs/outputs belonging to the channel of an application-specific module.

The IODDT associated with the counting modules are of T_ Unsigned_CPT_BMX and T_Signed_CPT_BMX types.

NOTE: IODDT variables can be created in two different ways:
- Using the I/O objects (see EcoStruxure™ Control Expert, Operating Modes) tab.
- Using the Data Editor (see EcoStruxure™ Control Expert, Operating Modes).

Language Object Types

Each IODDT contains a set of language objects allowing its operation to be controlled and checked.

There are two types of language objects:
- **Implicit Exchange Objects**: these objects are automatically exchanged on each cycle revolution of the task associated with the module.
- **Explicit Exchange Objects**: these objects are exchanged on the application's request, using explicit exchange instructions.

Implicit exchanges concern the inputs/outputs of the module (measurement results, information and commands). These exchanges enable the debugging of the counting modules.

Explicit exchanges enable the module to be set and diagnosed.
Implicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

An integrated application-specific interface or the addition of a module automatically enhances the language objects application used to program this interface or module. These objects correspond to the input/output images and software data of the module or integrated application-specific interface.

Reminders

The module inputs (%I and %IW) are updated in the PLC memory at the start of the task, the PLC being in RUN or STOP mode.

The outputs (%Q and %QW) are updated at the end of the task, only when the PLC is in RUN mode.

**NOTE:** When the task occurs in STOP mode, either of the following are possible, depending on the configuration selected:
- outputs are set to fallback position (fallback mode)
- outputs are maintained at their last value (maintain mode)

Figure

The following diagram shows the operating cycle of a PLC task (cyclical execution).

![Operating cycle diagram]

- Internal processing
- Acquisition of inputs
- Execution of the program
- Update of outputs

RUN

STOP
Explicit Exchange Language Objects Associated with the Application-Specific Function

Introduction

Explicit exchanges are performed at the user program's request using these instructions:
- READ_STS (read status words)
- WRITE_CMD (write command words)
- WRITE_PARAM (write adjustment parameters)
- READ_PARAM (read adjustment parameters)
- SAVE_PARAM (save adjustment parameters)
- RESTORE_PARAM (restore adjustment parameters)

For more details about instructions, refer to EcoStruxure™ Control Expert, I/O Management, Block Library.

These exchanges apply to a set of %MW objects of the same type (status, commands or parameters) that belong to a channel.

These objects can:
- provide information about the module (for example, type of error detected in a channel)
- have command control of the module (for example, switch command)
- define the module’s operating modes (save and restore adjustment parameters in the process of application)

NOTE: To avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (%MWr.m.c.0) of the IODDT associated to the channel before calling any EF addressing this channel.

NOTE: Explicit exchanges are not supported when X80 analog and digital I/O modules are configured through an eX80 adapter module (BMECRA31210) in a Quantum EIO configuration. You cannot set up a module's parameters from the PLC application during operation.
General Principle for Using Explicit Instructions

The diagram below shows the different types of explicit exchanges that can be made between the application and module.

Managing Exchanges

During an explicit exchange, check performance to see that the data is only taken into account when the exchange has been correctly executed.

To do this, two types of information is available:

- information concerning the exchange in progress (see page 182)
- the exchange report (see page 182)

The following diagram describes the management principle for an exchange.

NOTE: In order to avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS(%MWr.m.c.0) of the IODDT associated to the channel before calling any EF addressing this channel.
Management of Exchanges and Reports with Explicit Objects

At a Glance

When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information. IODDTs use two words to manage exchanges:

- **EXCH_STS (%MWr.m.c.0)**: exchange in progress
- **EXCH_RPT (%MWr.m.c.1)**: report

**NOTE:**
Depending on the localization of the module, the management of the explicit exchanges (%MW0.0.MOD.0.0 for example) will not be detected by the application:

- For in-rack modules, explicit exchanges are done immediately on the local PLC Bus and are finished before the end of the execution task. So, the **READ_STS**, for example, is finished when the %MW0.0.mod.0.0 bit is checked by the application.
- For remote bus (Fipio for example), explicit exchanges are not synchronous with the execution task, so the detection is possible by the application.

Illustration

The illustration below shows the different significant bits for managing exchanges:

- **Reconfiguration (bit 15)**
- **Adjustment (bit 2)**
- **Command (bit 1)**
- **Status (bit 0)**

Status parameters:
- **EXCH_RPT (%MWr.m.c.1)**
- **EXCH_STS (%MWr.m.c.0)**

Command parameters:
- **READ_STS**
- **WRITE_CMD**

Adjustment parameters:
- **WRITE_PARAM**
- **READ_PARAM**
- **SAVE_PARAM**
- **RESTORE_PARAM**
Description of Significant Bits

Each bit of the words EXCH_STS(%MWr.m.c.0) and EXCH_RPT(%MWr.m.c.1) is associated with a type of parameter:

- **Rank 0 bits** are associated with the status parameters:
  - The STS_IN_PROGR bit (%MWr.m.c.0.0) indicates whether a read request for the status words is in progress.
  - The STS_ERR bit (%MWr.m.c.1.0) specifies whether a read request for the status words is accepted by the module channel.

- **Rank 1 bits** are associated with the command parameters:
  - The CMD_IN_PROGR bit (%MWr.m.c.0.1) indicates whether command parameters are being sent to the module channel.
  - The CMD_ERR bit (%MWr.m.c.1.1) specifies whether the command parameters are accepted by the module channel.

- **Rank 2 bits** are associated with the adjustment parameters:
  - The ADJ_IN_PROGR bit (%MWr.m.c.0.2) indicates whether the adjustment parameters are being exchanged with the module channel (via WRITE_PARAM, READ_PARAM, SAVE_PARAM, RESTORE_PARAM).
  - The ADJ_ERR bit (%MWr.m.c.1.2) specifies whether the adjustment parameters are accepted by the module. If the exchange is correctly executed, the bit is set to 0.

- **Rank 15 bits** indicate a reconfiguration on channel c of the module from the console (modification of the configuration parameters + cold start-up of the channel).

- **The r, m and c bits** indicates the following elements:
  - The r bit represents the rack number.
  - The m bit represents the position of the module in the rack.
  - The c bit represents the channel number in the module.

**NOTE:** r represents the rack number, m the position of the module in the rack, while c represents the channel number in the module.

**NOTE:** Exchange and report words also exist at module level EXCH_STS(%MWr.m.MOD) and EXCH_RPT(%MWr.m.MOD.1) as per IODDT type T_GEN_MOD.
Example

Phase 1: Sending data by using the `WRITE_PARAM` instruction

<table>
<thead>
<tr>
<th>PLC memory</th>
<th>I/O module memory or integrated specific-application function memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status parameters</td>
<td>Status parameters</td>
</tr>
<tr>
<td>Command parameters</td>
<td>Command parameters</td>
</tr>
<tr>
<td>Adjustment parameters</td>
<td>Adjustment parameters</td>
</tr>
</tbody>
</table>

When the instruction is scanned by the PLC, the *Exchange in progress* bit is set to 1 in `%MWr.m.c.`

Phase 2: Analysis of the data by the I/O module and report

<table>
<thead>
<tr>
<th>PLC memory</th>
<th>I/O module memory or integrated specific-application function memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status parameters</td>
<td>Status parameters</td>
</tr>
<tr>
<td>Command parameters</td>
<td>Command parameters</td>
</tr>
<tr>
<td>Adjustment parameters</td>
<td>Adjustment parameters</td>
</tr>
</tbody>
</table>

When the data is exchanged between the PLC memory and the module, acknowledgement by the module is managed by the `ADJ_ERR` bit (%MWr.m.c.1.2).

This bit makes the following reports:
- 0: correct exchange
- 1: incorrect exchange

**NOTE:** There is no adjustment parameter at module level.
**Execution Indicators for an Explicit Exchange: EXCH_STS**

The table below shows the control bits of the explicit exchanges: `EXCH_STS(%MWr.m.c.0)`

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS_IN_PROGR</td>
<td>BOOL</td>
<td>R</td>
<td>Reading of channel status words in progress</td>
<td>%MWr.m.c.0.0</td>
</tr>
<tr>
<td>CMD_IN_PROGR</td>
<td>BOOL</td>
<td>R</td>
<td>Command parameters exchange in progress</td>
<td>%MWr.m.c.0.1</td>
</tr>
<tr>
<td>ADJ_IN_PROGR</td>
<td>BOOL</td>
<td>R</td>
<td>Adjust parameters exchange in progress</td>
<td>%MWr.m.c.0.2</td>
</tr>
<tr>
<td>RECONF_IN_PROGR</td>
<td>BOOL</td>
<td>R</td>
<td>Reconfiguration of the module in progress</td>
<td>%MWr.m.c.0.15</td>
</tr>
</tbody>
</table>

**NOTE:** If the module is not present or is disconnected, explicit exchange objects (e.g., `READ_STS` for example) are not sent to the module (`STS_IN_PROG (%MWr.m.c.0.0) = 0`), but the words are refreshed.

**Explicit Exchange Report: EXCH_RPT**

The table below shows the report bits: `EXCH_RPT(%MWr.m.c.1)`

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error detected while reading channel status words (1 = detected error)</td>
<td>%MWr.m.c.1.0</td>
</tr>
<tr>
<td>CMD_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error detected during a command parameter exchange (1 = detected error)</td>
<td>%MWr.m.c.1.1</td>
</tr>
<tr>
<td>ADJ_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error detected during an adjust parameter exchange (1 = detected error)</td>
<td>%MWr.m.c.1.2</td>
</tr>
<tr>
<td>RECONF_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error detected during reconfiguration of the channel (1 = detected error)</td>
<td>%MWr.m.c.1.15</td>
</tr>
</tbody>
</table>
Counting Module Use

The following table describes the steps realized between a counting module and the system after a power-on.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Power on.</td>
</tr>
<tr>
<td>2</td>
<td>The system sends the configuration parameters.</td>
</tr>
<tr>
<td>3</td>
<td>The system sends the adjust parameters by WRITE_PARAM method. <strong>Note:</strong> When the operation is finished, the bit %MWm.c.0.2 switches to 0.</td>
</tr>
</tbody>
</table>

If, in the beginning of your application, you use a WRITE_PARAM command, wait until the bit %MWm.c.0.2 switches to 0.
Section 13.2
Language Objects and IODDT Associated with the Counting Function of the BMX EHC xxxx Modules.

Subject of this Section
This section presents the language objects and IODDTs associated with the counting function of BMX EHC xxxx modules.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details of Implicit Exchange Objects for the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs</td>
<td>185</td>
</tr>
<tr>
<td>Details of the Explicit Exchange Objects for the T_CPT_BMX-type IODDT</td>
<td>190</td>
</tr>
</tbody>
</table>
Details of Implicit Exchange Objects for the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs

At a Glance

The tables below present the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs implicit exchange objects which are applicable to all BMX EHC counting modules.

Counter Value and Sensor Values

The table below presents the various IODDT implicit exchange objects:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTER_CURRENT_VALUE</td>
<td>DINT</td>
<td>R</td>
<td>Current counter value</td>
<td>%IDr.m.c.2</td>
</tr>
<tr>
<td>CAPT_0_VALUE</td>
<td>DINT</td>
<td>R</td>
<td>Counter value when captured in register 0</td>
<td>%IDr.m.c.4</td>
</tr>
<tr>
<td>CAPT_1_VALUE</td>
<td>DINT</td>
<td>R</td>
<td>Counter value when captured in register 1</td>
<td>%IDr.m.c.6</td>
</tr>
<tr>
<td>COUNTER_VALUE</td>
<td>DINT</td>
<td>R</td>
<td>Current counter value during event</td>
<td>%IDr.m.c.12</td>
</tr>
<tr>
<td>CAPT_0_VAL</td>
<td>DINT</td>
<td>R</td>
<td>Capture value 0</td>
<td>%IDr.m.c.14</td>
</tr>
<tr>
<td>CAPT_1_VAL</td>
<td>DINT</td>
<td>R</td>
<td>Capture value 1</td>
<td>%IDr.m.c.16</td>
</tr>
</tbody>
</table>

%Ir.m.c.d Word

The table below presents the meanings of the %Ir.m.c.d words:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH_ERROR</td>
<td>BOOL</td>
<td>R</td>
<td>Channel error</td>
<td>%Ir.m.c.ERR</td>
</tr>
<tr>
<td>OUTPUT_0_Echo</td>
<td>BOOL</td>
<td>R</td>
<td>Logical state of output 0</td>
<td>%Ir.m.c.0</td>
</tr>
<tr>
<td>OUTPUT_1_Echo</td>
<td>BOOL</td>
<td>R</td>
<td>Logical state of output 1</td>
<td>%Ir.m.c.1</td>
</tr>
<tr>
<td>OUTPUT_BLOCK_0</td>
<td>BOOL</td>
<td>R</td>
<td>State of output block 0</td>
<td>%Ir.m.c.2</td>
</tr>
<tr>
<td>OUTPUT_BLOCK_1</td>
<td>BOOL</td>
<td>R</td>
<td>State of output block 1</td>
<td>%Ir.m.c.3</td>
</tr>
<tr>
<td>INPUT_A</td>
<td>BOOL</td>
<td>R</td>
<td>Physical state of IN_A input</td>
<td>%Ir.m.c.4</td>
</tr>
<tr>
<td>INPUT_B</td>
<td>BOOL</td>
<td>R</td>
<td>Physical state of IN_B input</td>
<td>%Ir.m.c.5</td>
</tr>
<tr>
<td>INPUT_SYNC</td>
<td>BOOL</td>
<td>R</td>
<td>Physical state of the IN_SYNC input (or IN_AUX)</td>
<td>%Ir.m.c.6</td>
</tr>
<tr>
<td>INPUT_EN</td>
<td>BOOL</td>
<td>R</td>
<td>Physical state of IN_EN input (enable)</td>
<td>%Ir.m.c.7</td>
</tr>
<tr>
<td>INPUT_REF</td>
<td>BOOL</td>
<td>R</td>
<td>Physical state of the IN_REF input (preset)</td>
<td>%Ir.m.c.8</td>
</tr>
<tr>
<td>INPUT_CAPT</td>
<td>BOOL</td>
<td>R</td>
<td>Physical state of IN_CAP input (capture)</td>
<td>%Ir.m.c.9</td>
</tr>
</tbody>
</table>
## Counter Status, %IWr.m.c.0 Word

The following table presents the meanings of the bits of the %IWr.m.c.0 status word:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN</td>
<td>BOOL</td>
<td>R</td>
<td>The counter operates in counting mode only</td>
<td>%IWr.m.c.0.0</td>
</tr>
<tr>
<td>MODULO_FLAG</td>
<td>BOOL</td>
<td>R</td>
<td>Flag set to 1 by a modulo switch event</td>
<td>%IWr.m.c.0.1</td>
</tr>
<tr>
<td>SYNC_REF_FLAG</td>
<td>BOOL</td>
<td>R</td>
<td>Flag set to 1 by a preset or synchronization event</td>
<td>%IWr.m.c.0.2</td>
</tr>
<tr>
<td>VALIDITY</td>
<td>BOOL</td>
<td>R</td>
<td>The current numerical value is valid</td>
<td>%IWr.m.c.0.3</td>
</tr>
<tr>
<td>HIGH_LIMIT</td>
<td>BOOL</td>
<td>R</td>
<td>The current numerical value is locked at the upper threshold value</td>
<td>%IWr.m.c.0.4</td>
</tr>
<tr>
<td>LOW_LIMIT</td>
<td>BOOL</td>
<td>R</td>
<td>The current numerical value is locked at the lower threshold value</td>
<td>%IWr.m.c.0.5</td>
</tr>
</tbody>
</table>

## Comparison Status, %IWr.m.c.1 Word

The following table presents the meanings of the bits of the %IWr.m.c.1 status word:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTER_LOW</td>
<td>BOOL</td>
<td>R</td>
<td>Current counter value less than lower threshold (QDr.m.c.2)</td>
<td>%IWr.m.c.1.0</td>
</tr>
<tr>
<td>COUNTER_WIN</td>
<td>BOOL</td>
<td>R</td>
<td>Current counter value is between lower threshold (QDr.m.c.2) and upper threshold (QDr.m.c.4)</td>
<td>%IWr.m.c.1.1</td>
</tr>
<tr>
<td>COUNTER_HIGH</td>
<td>BOOL</td>
<td>R</td>
<td>Current counter value greater than upper threshold (QDr.m.c.4)</td>
<td>%IWr.m.c.1.2</td>
</tr>
<tr>
<td>CAPT_0_LOW</td>
<td>BOOL</td>
<td>R</td>
<td>Value captured in register 0 is less than lower threshold (QDr.m.c.2)</td>
<td>%IWr.m.c.1.3</td>
</tr>
<tr>
<td>CAPT_0_WIN</td>
<td>BOOL</td>
<td>R</td>
<td>Value captured in register 0 is between lower threshold (QDr.m.c.2) and upper threshold (QDr.m.c.4)</td>
<td>%IWr.m.c.1.4</td>
</tr>
<tr>
<td>CAPT_0_HIGH</td>
<td>BOOL</td>
<td>R</td>
<td>Value captured in register 0 is greater than upper threshold (QDr.m.c.4)</td>
<td>%IWr.m.c.1.5</td>
</tr>
<tr>
<td>CAPT_1_LOW</td>
<td>BOOL</td>
<td>R</td>
<td>Value captured in register 1 is less than lower threshold (QDr.m.c.2)</td>
<td>%IWr.m.c.1.6</td>
</tr>
<tr>
<td>CAPT_1_WIN</td>
<td>BOOL</td>
<td>R</td>
<td>Value captured in register 1 is between lower threshold (QDr.m.c.2) and upper threshold (QDr.m.c.4)</td>
<td>%IWr.m.c.1.7</td>
</tr>
<tr>
<td>CAPT_1_HIGH</td>
<td>BOOL</td>
<td>R</td>
<td>Value captured in register 1 is greater than upper threshold (QDr.m.c.4)</td>
<td>%IWr.m.c.1.8</td>
</tr>
</tbody>
</table>
Event Sources, %IWr.m.c.10 Word

The following table presents the meanings of the bits of the %IWr.m.c.10 word:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVT_SOURCES</td>
<td>INT</td>
<td>R</td>
<td>Event sources field</td>
<td>%IWr.m.c.10</td>
</tr>
<tr>
<td>EVT_RUN</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to start of counter.</td>
<td>%IWr.m.c.10.0</td>
</tr>
<tr>
<td>EVT_MODULO</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to modulo switch</td>
<td>%IWr.m.c.10.1</td>
</tr>
<tr>
<td>EVT_SYNC_PRESET</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to synchronization or preset</td>
<td>%IWr.m.c.10.2</td>
</tr>
<tr>
<td>EVT_COUNTER_LOW</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to counter value being less than lower threshold</td>
<td>%IWr.m.c.10.3</td>
</tr>
<tr>
<td>EVT_COUNTER_WINDOW</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to counter value being between the two thresholds</td>
<td>%IWr.m.c.10.4</td>
</tr>
<tr>
<td>EVT_COUNTER_HIGH</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to counter value being greater than upper threshold</td>
<td>%IWr.m.c.10.5</td>
</tr>
<tr>
<td>EVT_CAPT_0</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to capture function 0</td>
<td>%IWr.m.c.10.6</td>
</tr>
<tr>
<td>EVT_CAPT_1</td>
<td>BOOL</td>
<td>R</td>
<td>Event due to capture function 1</td>
<td>%IWr.m.c.10.7</td>
</tr>
<tr>
<td>EVT_OVERRUN</td>
<td>BOOL</td>
<td>R</td>
<td>Warning: lost event(s)</td>
<td>%IWr.m.c.10.8</td>
</tr>
</tbody>
</table>

Output Thresholds and Frequency

The table below presents the various IODDT implicit exchange objects:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOWER_TH_VALUE</td>
<td>DINT</td>
<td>R/W</td>
<td>Lower threshold value</td>
<td>%QDr.m.c.2</td>
</tr>
<tr>
<td>UPPER_TH_VALUE</td>
<td>DINT</td>
<td>R/W</td>
<td>Upper threshold value</td>
<td>%QDr.m.c.4</td>
</tr>
<tr>
<td>PWM_FREQUENCY</td>
<td>DINT</td>
<td>R/W</td>
<td>Output frequency value (unit = 0.1 Hz)</td>
<td>%QDr.m.c.6</td>
</tr>
<tr>
<td>PWM_DUTY</td>
<td>INT</td>
<td>R/W</td>
<td>Duty cycle value of the output frequency (unit = 5%)</td>
<td>%QDr.m.c.8</td>
</tr>
</tbody>
</table>
%Qr.m.c.d Words

The following table presents the meanings of the bits of the %Qr.m.c.d words:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT_0</td>
<td>BOOL</td>
<td>R/W</td>
<td>Forces OUTPUT_0 to level 1</td>
<td>%Qr.m.c.0</td>
</tr>
<tr>
<td>OUTPUT_1</td>
<td>BOOL</td>
<td>R/W</td>
<td>Forces OUTPUT_1 to level 1</td>
<td>%Qr.m.c.1</td>
</tr>
<tr>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>Implementation of output 0 function block</td>
<td>%Qr.m.c.2</td>
</tr>
<tr>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>Implementation of output 1 function block</td>
<td>%Qr.m.c.3</td>
</tr>
<tr>
<td>FORCE_SYNC</td>
<td>BOOL</td>
<td>R/W</td>
<td>Counting function synchronization and start</td>
<td>%Qr.m.c.4</td>
</tr>
<tr>
<td>FORCE_REF</td>
<td>BOOL</td>
<td>R/W</td>
<td>Set to preset counter value</td>
<td>%Qr.m.c.5</td>
</tr>
<tr>
<td>FORCE_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>Implementation of counter</td>
<td>%Qr.m.c.6</td>
</tr>
<tr>
<td>FORCE_RESET</td>
<td>BOOL</td>
<td>R/W</td>
<td>Reset counter</td>
<td>%Qr.m.c.7</td>
</tr>
<tr>
<td>SYNC_RESET</td>
<td>BOOL</td>
<td>R/W</td>
<td>Reset SYNC_REF_FLAG</td>
<td>%Qr.m.c.8</td>
</tr>
<tr>
<td>MODULO_RESET</td>
<td>BOOL</td>
<td>R/W</td>
<td>Reset MODULO_FLAG</td>
<td>%Qr.m.c.9</td>
</tr>
</tbody>
</table>

FUNCTIONS_ENABLING, %Wr.m.c.0 Word

The following table presents the meanings of the bits of the %Wr.m.c.0 words:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALID_SYNC</td>
<td>BOOL</td>
<td>R/W</td>
<td>Synchronization and start authorization for the counting function via the IN_SYNC input</td>
<td>%Wr.m.c.0.0</td>
</tr>
<tr>
<td>VALID_REF</td>
<td>BOOL</td>
<td>R/W</td>
<td>Operation authorization for the internal preset function</td>
<td>%Wr.m.c.0.1</td>
</tr>
<tr>
<td>VALID_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>Authorization of the counter enable via the IN_EN input</td>
<td>%Wr.m.c.0.2</td>
</tr>
<tr>
<td>VALID_CAPT_0</td>
<td>BOOL</td>
<td>R/W</td>
<td>Capture authorization in the capture0 register</td>
<td>%Wr.m.c.0.3</td>
</tr>
<tr>
<td>VALID_CAPT_1</td>
<td>BOOL</td>
<td>R/W</td>
<td>Capture authorization in the capture1 register</td>
<td>%Wr.m.c.0.4</td>
</tr>
<tr>
<td>COMPARE_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>Comparators operation authorization</td>
<td>%Wr.m.c.0.5</td>
</tr>
<tr>
<td>COMPARE_SUSPEND</td>
<td>BOOL</td>
<td>R/W</td>
<td>Comparator frozen at its last value</td>
<td>%Wr.m.c.0.6</td>
</tr>
</tbody>
</table>
## EVENT_SOURCES_ENABLING, %QWr.m.c.1 Word

The following table presents the meanings of the bits of the %QWr.m.c.1 words:

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVT_RUN_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call at start of the counting function</td>
<td>%QWr.m.c.1.0</td>
</tr>
<tr>
<td>EVT_MODULO_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call when there is a counter reversal</td>
<td>%QWr.m.c.1.1</td>
</tr>
<tr>
<td>EVT_REF_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call during counter synchronization or preset</td>
<td>%QWr.m.c.1.2</td>
</tr>
<tr>
<td>EVT_COUNTER_LOW_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call when the counter value is less than lower threshold</td>
<td>%QWr.m.c.1.3</td>
</tr>
<tr>
<td>EVT_COUNTER_WINDOW_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call when the counter is between the lower and upper threshold</td>
<td>%QWr.m.c.1.4</td>
</tr>
<tr>
<td>EVT_COUNTER_HIGH_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call when the counter value is greater than the upper threshold</td>
<td>%QWr.m.c.1.5</td>
</tr>
<tr>
<td>EVT_CAPT_0_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call during capture in register 0</td>
<td>%QWr.m.c.1.6</td>
</tr>
<tr>
<td>EVT_CAPT_1_ENABLE</td>
<td>BOOL</td>
<td>R/W</td>
<td>EVENT task call during capture in register 1</td>
<td>%QWr.m.c.1.7</td>
</tr>
</tbody>
</table>
Details of the Explicit Exchange Objects for the T_CPT_BMX-type IODDT

At a Glance

This section presents the explicit exchange objects for the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs which are applicable to all BMX EHC counting modules. They include word type objects whose bits have a specific meaning. These objects are described in detail below.

Sample variable declaration: T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDT_VAR1.

NOTE:
- In general, the meaning of the bits is given for bit status 1.
- Not all bits are used.

Exchange Status: EXCH_STS

The table below shows the meaning of channel exchange status bits from the EXCH_STS channel (%MWr.m.c.0).

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS_IN_PROG</td>
<td>BOOL</td>
<td>R</td>
<td>Status parameter read in progress</td>
<td>%MWr.m.c.0.0</td>
</tr>
<tr>
<td>ADJ_IN_PROG</td>
<td>BOOL</td>
<td>R</td>
<td>Adjust parameter exchange in progress</td>
<td>%MWr.m.c.0.2</td>
</tr>
<tr>
<td>RECONF_IN_PROG</td>
<td>BOOL</td>
<td>R</td>
<td>Reconfiguration in progress</td>
<td>%MWr.m.c.0.15</td>
</tr>
</tbody>
</table>

Channel Report: EXCH_RPT

The following table presents the meanings of the report bits of the EXCH_RPT channel (%MWr.m.c.1).

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error while reading channel status</td>
<td>%MWr.m.c.1.0</td>
</tr>
<tr>
<td>ADJ_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error while adjusting the channel</td>
<td>%MWr.m.c.1.2</td>
</tr>
<tr>
<td>RECONF_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Error while reconfiguring the channel</td>
<td>%MWr.m.c.1.15</td>
</tr>
</tbody>
</table>
Channel Error: CH_FLT

The table below presents the meaning of the error bits on the CH_FLT channel (%MWr.m.c.2).

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTERNAL_FLT_INPUTS</td>
<td>BOOL</td>
<td>R</td>
<td>External error at inputs</td>
<td>%MWr.m.c.2.0</td>
</tr>
<tr>
<td>EXTERNAL_FLT_OUTPUTS</td>
<td>BOOL</td>
<td>R</td>
<td>External error at outputs</td>
<td>%MWr.m.c.2.1</td>
</tr>
<tr>
<td>INTERNAL_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Internal error: channel inoperative</td>
<td>%MWr.m.c.2.4</td>
</tr>
<tr>
<td>CONF_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Hardware or software configuration error</td>
<td>%MWr.m.c.2.5</td>
</tr>
<tr>
<td>COM_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Bus Communication error</td>
<td>%MWr.m.c.2.6</td>
</tr>
<tr>
<td>APPLI_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Application error</td>
<td>%MWr.m.c.2.7</td>
</tr>
</tbody>
</table>

Channel Error: %MWr.m.c.3

The table below presents the meaning of the error bits on the %MWr.m.c.3 word.

<table>
<thead>
<tr>
<th>Standard symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Language object</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSOR_SUPPLY</td>
<td>BOOL</td>
<td>R</td>
<td>Low input power supply for the sensors</td>
<td>%MWr.m.c.3.2</td>
</tr>
<tr>
<td>ACTUATOR_SUPPLY_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Output power supply failure</td>
<td>%MWr.m.c.3.3</td>
</tr>
<tr>
<td>SHORT_CIRCUIT_OUT_0</td>
<td>BOOL</td>
<td>R</td>
<td>Short circuit on output 0</td>
<td>%MWr.m.c.3.4</td>
</tr>
<tr>
<td>SHORT_CIRCUIT_OUT_1</td>
<td>BOOL</td>
<td>R</td>
<td>Short circuit on output 1</td>
<td>%MWr.m.c.3.5</td>
</tr>
</tbody>
</table>
Section 13.3
The IODDT Type T_GEN_MOD Applicable to All Modules

Details of the Language Objects of the IODDT of Type T_GEN_MOD

Introduction
The Modicon X80 modules have an associated IODDT of type T_GEN_MOD.

Observations
In general, the meaning of the bits is given for bit status 1. In specific cases an explanation is given for each status of the bit.
Some bits are not used.

List of Objects
The table below presents the objects of the IODDT.

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Access</th>
<th>Meaning</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOD_ERROR</td>
<td>BOOL</td>
<td>R</td>
<td>Module detected error bit</td>
<td>%Ir.m.MOD.ERR</td>
</tr>
<tr>
<td>EXCH_STS</td>
<td>INT</td>
<td>R</td>
<td>Module exchange control word</td>
<td>%MWr.m.MOD.0</td>
</tr>
<tr>
<td>STS_IN_PROGR</td>
<td>BOOL</td>
<td>R</td>
<td>Reading of status words of the module in progress</td>
<td>%MWr.m.MOD.0.0</td>
</tr>
<tr>
<td>EXCH_RPT</td>
<td>INT</td>
<td>R</td>
<td>Exchange report word</td>
<td>%MWr.m.MOD.1</td>
</tr>
<tr>
<td>STS_ERR</td>
<td>BOOL</td>
<td>R</td>
<td>Event when reading module status words</td>
<td>%MWr.m.MOD.1.0</td>
</tr>
<tr>
<td>MOD_FAIL</td>
<td>INT</td>
<td>R</td>
<td>Internal detected errors word of the module</td>
<td>%MWr.m.MOD.2</td>
</tr>
<tr>
<td>MOD_FAIL_EXT</td>
<td>BOOL</td>
<td>R</td>
<td>Internal detected error, module unserviceable</td>
<td>%MWr.m.MOD.2.8</td>
</tr>
<tr>
<td>CH_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Inoperative channel(s)</td>
<td>%MWr.m.MOD.2.1</td>
</tr>
<tr>
<td>BLK</td>
<td>BOOL</td>
<td>R</td>
<td>Terminal block incorrectly wired</td>
<td>%MWr.m.MOD.2.2</td>
</tr>
<tr>
<td>CONF_FLT</td>
<td>BOOL</td>
<td>R</td>
<td>Hardware or software configuration anomaly</td>
<td>%MWr.m.MOD.2.5</td>
</tr>
<tr>
<td>NO_MOD</td>
<td>BOOL</td>
<td>R</td>
<td>Module missing or inoperative</td>
<td>%MWr.m.MOD.2.6</td>
</tr>
<tr>
<td>EXT_MODFLT</td>
<td>BOOL</td>
<td>R</td>
<td>Internal detected errors word of the module</td>
<td>%MWr.m.MOD.2.7</td>
</tr>
<tr>
<td>MOD_FAIL_EXT</td>
<td>BOOL</td>
<td>R</td>
<td>Internal detected error, module unserviceable</td>
<td>%MWr.m.MOD.2.8</td>
</tr>
<tr>
<td>CH_FLT_EXT</td>
<td>BOOL</td>
<td>R</td>
<td>Inoperative channel(s)</td>
<td>%MWr.m.MOD.2.9</td>
</tr>
<tr>
<td>BLK_EXT</td>
<td>BOOL</td>
<td>R</td>
<td>Terminal block incorrectly wired</td>
<td>%MWr.m.MOD.2.10</td>
</tr>
<tr>
<td>Standard Symbol</td>
<td>Type</td>
<td>Access</td>
<td>Meaning</td>
<td>Address</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>--------</td>
<td>----------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>CONF_FLT_EXT</td>
<td>BOOL</td>
<td>R</td>
<td>Hardware or software configuration anomaly</td>
<td>%MWr.m.MOD.2.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Fipio extension only)</td>
<td></td>
</tr>
<tr>
<td>NO_MOD_EXT</td>
<td>BOOL</td>
<td>R</td>
<td>Module missing or inoperative (Fipio extension only)</td>
<td>%MWr.m.MOD.2.14</td>
</tr>
</tbody>
</table>
Section 13.4
Device DDTs Associated with the Counting Function of the BMX EHC xxxx Modules.

Subject of this Section
This section presents the Device DDTs associated with the counting function of BMX EHC xxxx modules.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter Device DDT</td>
<td>195</td>
</tr>
<tr>
<td>MODFLT Byte Description</td>
<td>203</td>
</tr>
</tbody>
</table>
Counter Device DDT

Introduction

This topic describes the device DDT for the Modicon X80 counter module, the instance default naming is described in Device DDT Instance Naming Rule (see EcoStruxure™ Control Expert, Program Languages and Structure, Reference Manual).

Regarding the device DDT, its name contains the following information:

- **platform** with:
  - M for Modicon X80 module
- **device type** (CPT for counter)
- **function** (STD for standard)
- **direction**:
  - IN
  - OUT
- **max channel** (2 or 8)

Example: For a Modicon X80 counter module with 2 standard inputs: T_M_CPT_STD_IN_2

Adjustment Parameter limitation

Adjustment parameters cannot be changed from the PLC application during operation (no support of READ_PARAM, WRITE_PARAM, SAVE_PARAM, RESTORE_PARAM) for:

- counter modules in a Quantum EIO
- counter modules in a M580 RIO

Modifying the adjustment parameters of a channel from Control Expert during a CCOTF operation causes the channel to be re-initialized.

The concerned parameters are:

- **PRESET_VALUE**
  - Preset value
- **CALIBRATION_FACTOR**
  - Calibration Factor
- **MODULO_VALUE**
  - Modulo value
- **SLACK_VAL** (Hysteresis)
  - Offset value

List of Implicit Device DDT

The following table shows the list of device DDT and their X80 modules:

<table>
<thead>
<tr>
<th>Device DDT</th>
<th>Modicon X80 modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_M_CPT_STD_IN_2</td>
<td>BMX EHC 0200</td>
</tr>
<tr>
<td>T_M_CPT_STD_IN_8</td>
<td>BMX EHC 0800</td>
</tr>
</tbody>
</table>
## Implicit Device DDT Description

The following table shows the T_M_CPT_STD_IN_x status word bits:

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOD_HEALTH</td>
<td>BOOL</td>
<td>0 = the module has a detected error</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = the module is operating correctly</td>
<td></td>
</tr>
<tr>
<td>MOD_FLT</td>
<td>BYTE</td>
<td>internal detected errors byte (see page 203) of the module</td>
<td>read</td>
</tr>
<tr>
<td>CPT_CH_IN</td>
<td>ARRAY [0..x-1] of T_M_CPT_STD_CH_IN</td>
<td>Array of structure</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows the T_M_CPT_STD_CH_IN_x[0..x-1] status word bits:

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCT_TYPE</td>
<td>WORD</td>
<td>–</td>
<td>1 = Frequency</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 = EvtCounting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 = PeriodMeasuring</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 = Ratio1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 = Ratio2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 = OneShotCounter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 = ModuleLoopCounter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8 = FreeLargeCounter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9 = PulseWidthModulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10 = UpDownCounting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11 = DualPhaseCounting</td>
<td></td>
</tr>
<tr>
<td>CH.HEALTH</td>
<td>BOOL</td>
<td>–</td>
<td>0 = the channel has a detected error</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 = the channel is operating correctly</td>
<td></td>
</tr>
<tr>
<td>ST_OUTPUT_0_ECHO</td>
<td>EBOOL</td>
<td>–</td>
<td>logical state of output 0</td>
<td>read</td>
</tr>
<tr>
<td>ST_OUTPUT_1_ECHO</td>
<td>EBOOL</td>
<td>–</td>
<td>logical state of output 1</td>
<td>read</td>
</tr>
<tr>
<td>ST_OUTPUT_BLOCK_0</td>
<td>EBOOL</td>
<td>–</td>
<td>status of physical counting output block 0</td>
<td>read</td>
</tr>
<tr>
<td>ST_OUTPUT_BLOCK_1</td>
<td>EBOOL</td>
<td>–</td>
<td>status of physical counting output block 1</td>
<td>read</td>
</tr>
</tbody>
</table>

(1) Signed application specific function (ASF) must be used
(2) Unsigned application specific function (ASF) must be used
<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_INPUT_A</td>
<td>EBOOL</td>
<td>–</td>
<td>status of physical counting input A</td>
<td>read</td>
</tr>
<tr>
<td>ST_INPUT_B</td>
<td>EBOOL</td>
<td>–</td>
<td>status of physical counting input B</td>
<td>read</td>
</tr>
<tr>
<td>ST_INPUT_SYNC</td>
<td>EBOOL</td>
<td>–</td>
<td>physical state of the IN_SYNC input (or IN_AUX)</td>
<td>read</td>
</tr>
<tr>
<td>ST_INPUT_EN</td>
<td>EBOOL</td>
<td>–</td>
<td>physical state of IN_EN input (enable)</td>
<td>read</td>
</tr>
<tr>
<td>ST_INPUT_REF</td>
<td>EBOOL</td>
<td>–</td>
<td>physical state of IN_REF input (preset)</td>
<td>read</td>
</tr>
<tr>
<td>ST_INPUT_CAPT</td>
<td>EBOOL</td>
<td>–</td>
<td>physical state of IN_CAP input (capture)</td>
<td>read</td>
</tr>
<tr>
<td>COUNTER_STATUS</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUN</td>
<td>BOOL</td>
<td>0</td>
<td>the counter operates in counting mode only</td>
<td>read</td>
</tr>
<tr>
<td>MODULO_FLAG</td>
<td>BOOL</td>
<td>1</td>
<td>flag set to 1 by a modulo switch event</td>
<td>read</td>
</tr>
<tr>
<td>SYNC_REF_FLAG</td>
<td>BOOL</td>
<td>2</td>
<td>flag set to 1 by a preset or synchronization event</td>
<td>read</td>
</tr>
<tr>
<td>VALIDITY</td>
<td>BOOL</td>
<td>3</td>
<td>the current numerical value is valid</td>
<td>read</td>
</tr>
<tr>
<td>HIGH_LIMIT</td>
<td>BOOL</td>
<td>4</td>
<td>the current numerical value is locked at the upper threshold value</td>
<td>read</td>
</tr>
<tr>
<td>LOW_LIMIT</td>
<td>BOOL</td>
<td>5</td>
<td>the current numerical value is locked at the lower threshold value</td>
<td>read</td>
</tr>
</tbody>
</table>

(1) Signed application specific function (ASF) must be used
(2) Unsigned application specific function (ASF) must be used
<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPARE_STATUS [INT]</td>
<td>COUNTER_LOW</td>
<td>BOOL 0</td>
<td>current counter value less than lower threshold (LOWER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>COUNTER_WIN</td>
<td>BOOL 1</td>
<td>current counter value is between lower threshold (LOWER_TH_VALUE) and upper threshold (UPPER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>COUNTER_HIGH</td>
<td>BOOL 2</td>
<td>current counter value greater than upper threshold (UPPER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_0_LOW</td>
<td>BOOL 3</td>
<td>Value captured in register 0 is less than lower threshold (LOWER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_0_WIN</td>
<td>BOOL 4</td>
<td>Value captured in register 0 is between lower threshold (LOWER_TH_VALUE) and upper threshold (UPPER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_0_HIGH</td>
<td>BOOL 5</td>
<td>Value captured in register 0 is greater than upper threshold (UPPER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_1_LOW</td>
<td>BOOL 6</td>
<td>Value captured in register 1 is less than lower threshold (LOWER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_1_WIN</td>
<td>BOOL 7</td>
<td>Value captured in register 1 is between lower threshold (LOWER_TH_VALUE) and upper threshold (UPPER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_1_HIGH</td>
<td>BOOL 8</td>
<td>Value captured in register 1 is greater than upper threshold (UPPER_TH_VALUE)</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>COUNTER_CURRENT_VALUE_S(1)</td>
<td>DINT</td>
<td>Current counter value during event</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_0_VALUE_S(1)</td>
<td>DINT</td>
<td>Value captured in register 0</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_1_VALUE_S(1)</td>
<td>DINT</td>
<td>Value captured in register 1</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>COUNTER_CURRENT_VALUE_US(2)</td>
<td>UDINT</td>
<td>Current counter value during event</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CAPT_0_VALUE_US(2)</td>
<td>UDINT</td>
<td>Value captured in register 0</td>
<td>read</td>
</tr>
</tbody>
</table>

(1) Signed application specific function (ASF) must be used
(2) Unsigned application specific function (ASF) must be used
<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPT_1_VALUE_US(2)</td>
<td>UDINT</td>
<td>–</td>
<td>Value captured in register 1</td>
<td>read</td>
</tr>
<tr>
<td>OUTPUT_0</td>
<td>EBOOL</td>
<td>–</td>
<td>forces OUTPUT_0 to level 1</td>
<td>read / write</td>
</tr>
<tr>
<td>OUTPUT_1</td>
<td>EBOOL</td>
<td>–</td>
<td>forces OUTPUT_1 to level 1</td>
<td>read / write</td>
</tr>
<tr>
<td>OUTPUT_BLOCK_0_ENABLE</td>
<td>EBOOL</td>
<td>–</td>
<td>implementation of output 0 function block</td>
<td>read / write</td>
</tr>
<tr>
<td>OUTPUT_BLOCK_1_ENABLE</td>
<td>EBOOL</td>
<td>–</td>
<td>implementation of output 1 function block</td>
<td>read / write</td>
</tr>
<tr>
<td>FORCE_SYNC</td>
<td>EBOOL</td>
<td>–</td>
<td>counting function synchronization and start</td>
<td>read / write</td>
</tr>
<tr>
<td>FORCE_REF</td>
<td>EBOOL</td>
<td>–</td>
<td>set to preset counter value</td>
<td>read / write</td>
</tr>
<tr>
<td>FORCE_ENABLE</td>
<td>EBOOL</td>
<td>–</td>
<td>implementation of counter</td>
<td>read / write</td>
</tr>
<tr>
<td>FORCE_RESET</td>
<td>EBOOL</td>
<td>–</td>
<td>reset counter</td>
<td>read / write</td>
</tr>
<tr>
<td>SYNC_RESET</td>
<td>EBOOL</td>
<td>–</td>
<td>reset SYNC_REF_FLAG</td>
<td>read / write</td>
</tr>
<tr>
<td>MODULO_RESET</td>
<td>EBOOL</td>
<td>–</td>
<td>reset MODULO_FLAG</td>
<td>read / write</td>
</tr>
<tr>
<td>FUNCTIONS_ENABLING [INT]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALID_SYNC</td>
<td>BOOL</td>
<td>0</td>
<td>synchronization and start authorization for the counting function via the IN_SYNC input</td>
<td>read / write</td>
</tr>
<tr>
<td>VALID_REF</td>
<td>BOOL</td>
<td>1</td>
<td>operation authorization for the internal preset function</td>
<td>read / write</td>
</tr>
<tr>
<td>VALID_ENABLE</td>
<td>BOOL</td>
<td>2</td>
<td>authorization of the counter enable via the IN_EN input</td>
<td>read / write</td>
</tr>
<tr>
<td>VALID_CAPT_0</td>
<td>BOOL</td>
<td>3</td>
<td>capture authorization in the capture 0 register</td>
<td>read / write</td>
</tr>
<tr>
<td>VALID_CAPT_1</td>
<td>BOOL</td>
<td>4</td>
<td>capture authorization in the capture 1 register</td>
<td>read / write</td>
</tr>
<tr>
<td>COMPARE_ENABLE</td>
<td>BOOL</td>
<td>5</td>
<td>comparators operation authorization</td>
<td>read / write</td>
</tr>
<tr>
<td>COMPARE_SUSPEND</td>
<td>BOOL</td>
<td>6</td>
<td>comparator frozen at its last value</td>
<td>read / write</td>
</tr>
<tr>
<td>LOWER_TH_VALUE_S(1)</td>
<td>DINT</td>
<td>–</td>
<td>lower threshold value</td>
<td>read / write</td>
</tr>
</tbody>
</table>

(1) Signed application specific function (ASF) must be used
(2) Unsigned application specific function (ASF) must be used
Here below is all the signed ASF that must be used with a counter BMX EHC 0200:

- Free Large counter Mode
- Ratio 1
- Ratio 2

Here below is all the unsigned ASF that must be used with a counter BMX EHC 0200:

- Event Counting Mode
- Frequency Mode
- Modulo Loop Counter Mode
- One Shot Counter Mode
- Period Measuring Mode
- Pulse Width Modulation Mode

Here below is all the signed ASF that must be used with a counter BMX EHC 0800:

- Up Down Counting Mode

Here below is all the unsigned ASF that must be used with a counter BMX EHC 0800:

- Event Counting Mode
- Frequency Mode
- Modulo Loop Counter Mode
- One Shot Counter Mode

---

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER_TH_VALUE_S&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>DINT</td>
<td>–</td>
<td>upper threshold value</td>
<td>read / write</td>
</tr>
<tr>
<td>PWM_FREQUENCY_S&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>DINT</td>
<td>–</td>
<td>output frequency value (unit = 0.1 Hz)</td>
<td>read / write</td>
</tr>
<tr>
<td>LOWER_TH_VALUE_US&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>UDINT</td>
<td>–</td>
<td>lower threshold value</td>
<td>read / write</td>
</tr>
<tr>
<td>UPPER_TH_VALUE_US&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>UDINT</td>
<td>–</td>
<td>upper threshold value</td>
<td>read / write</td>
</tr>
<tr>
<td>PWM_FREQUENCY_US&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>UDINT</td>
<td>–</td>
<td>output frequency value (unit = 0.1 Hz)</td>
<td>read / write</td>
</tr>
<tr>
<td>PWM_DUTY</td>
<td>INT</td>
<td>–</td>
<td>duty cycle value of the output frequency (unit = 5%)</td>
<td>read / write</td>
</tr>
</tbody>
</table>

<sup>(1)</sup> Signed application specific function (ASF) must be used

<sup>(2)</sup> Unsigned application specific function (ASF) must be used
### Use and Description of DDT for Explicit Exchange

The following table shows the Derived Data Type (DDT) used for the variables connected to dedicated EFB parameter to perform an explicit exchange:

<table>
<thead>
<tr>
<th>DDT</th>
<th>Description</th>
<th>Depending on the module location, the DDT can be connected to the <strong>STS</strong> output parameter of the EFB:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T_M_CPT_STD_CH_STS</strong></td>
<td>Structure to read the channel status of a counting module.</td>
<td>● READ_STS_QX when the module is located in Quantum EIO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● READ_STS_MX when the module is located in a M580 local rack or in M580 RIO drops.</td>
</tr>
<tr>
<td><strong>T_M_SIGN_CPT_STD_CH_PRM</strong></td>
<td>Structure for adjustment parameters of a channel of a counting module (signed application specific function) in a M580 local rack.</td>
<td>The DDT can be connected to the <strong>PARAM</strong> output parameter of the EFB:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● READ_PARAM_MX to read module parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● WRITE_PARAM_MX to write module parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● SAVE_PARAM_MX to save module parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>● RESTORE_PARAM_MX to restore the new parameters of the module.</td>
</tr>
<tr>
<td><strong>T_M_UNSIGN_CPT_STD_CH_PRM</strong></td>
<td>Structure for adjustment parameters of a channel of a counting module (unsigned application specific function) in a M580 local rack.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** Targeted channel address (**ADDR**) can be managed with ADDMX *(see EcoStruxure™ Control Expert, Communication, Block Library)* EF (connect the output parameter **OUT** to the input parameter **ADDR** of the communication functions).
The following table shows the structure of the T_M_CPT_STD_CH_STS DDT:

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH_FLT [INT]</td>
<td>EXTERNAL_FLT_INPUTS</td>
<td>BOOL 0</td>
<td>external detected error at inputs</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>EXTERNAL_FLT_OUTPUTS</td>
<td>BOOL 1</td>
<td>external detected error at outputs</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>INTERNAL_FLT</td>
<td>BOOL 4</td>
<td>internal detected error: channel inoperative</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>CONF_FLT</td>
<td>BOOL 5</td>
<td>hardware or software configuration detected error</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>COM_FLT</td>
<td>BOOL 6</td>
<td>bus communication detected error</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>APPLI_FLT</td>
<td>BOOL 7</td>
<td>application detected error</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>COM_EVT_FLT</td>
<td>BOOL 8</td>
<td>communication event detected fault</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>OVR_EVT_CPU</td>
<td>BOOL 9</td>
<td>CPU overflow event</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>OVR_CPT_CH</td>
<td>BOOL 10</td>
<td>counter channel overflow</td>
<td>read</td>
</tr>
<tr>
<td>CH_FLT_2 [INT]</td>
<td>SENSOR_SUPPLY</td>
<td>BOOL 2</td>
<td>low input power supply for the sensors</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>ACTUATOR_SUPPLY_FLT</td>
<td>BOOL 3</td>
<td>output power supply loss</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>SHORT_CIRCUIT_OUT_0</td>
<td>BOOL 4</td>
<td>short circuit on output 0</td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>SHORT_CIRCUIT_OUT_1</td>
<td>BOOL 5</td>
<td>short circuit on output 1</td>
<td>read</td>
</tr>
</tbody>
</table>

The following table shows the structure of the T_M_SIGN_CPT_STD_CH_PRM DDT:

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODULO_VALUE</td>
<td>DINT</td>
<td>–</td>
<td>Modulo value</td>
<td>read/write</td>
</tr>
<tr>
<td>PRESET_VALUE</td>
<td>DINT</td>
<td>–</td>
<td>Preset value</td>
<td>read/write</td>
</tr>
<tr>
<td>CALIBRATION_FACTOR</td>
<td>INT</td>
<td>–</td>
<td>Adjust the value from – 10 % to + 10 %, unit = 0.1 %</td>
<td>read/write</td>
</tr>
<tr>
<td>SLACK_VAL</td>
<td>INT</td>
<td>–</td>
<td>Hysteresis</td>
<td>read/write</td>
</tr>
</tbody>
</table>

The following table shows the structure of the T_M_UNSIGN_CPT_STD_CH_PRM DDT:

<table>
<thead>
<tr>
<th>Standard Symbol</th>
<th>Type</th>
<th>Bit</th>
<th>Meaning</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODULO_VALUE</td>
<td>UINT</td>
<td>–</td>
<td>Modulo value</td>
<td>read/write</td>
</tr>
<tr>
<td>PRESET_VALUE</td>
<td>UINT</td>
<td>–</td>
<td>Preset value</td>
<td>read/write</td>
</tr>
<tr>
<td>CALIBRATION_FACTOR</td>
<td>INT</td>
<td>–</td>
<td>Adjust the value from – 10 % to + 10 %, unit = 0.1 %</td>
<td>read/write</td>
</tr>
<tr>
<td>SLACK_VAL</td>
<td>INT</td>
<td>–</td>
<td>Hysteresis</td>
<td>read/write</td>
</tr>
</tbody>
</table>
## MOD_FLT Byte Description

### MOD_FLT Byte in Device DDT

#### MOD_FLT byte structure:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0   | MOD_FAIL | • 1: Internal detected error or module failure detected.  
     |          | • 0: No detected error                                                      |
| 1   | CH_FLT   | • 1: Inoperative channels.  
     |          | • 0: Channels are operative.                                               |
| 2   | BLK      | • 1: Terminal block detected error.  
     |          | • 0: No detected error                                                     |
|     |          | **NOTE:** This bit may not be managed.                                     |
| 3   | –        | • 1: Module in self-test.  
     |          | • 0: Module not in self-test                                               |
|     |          | **NOTE:** This bit may not be managed.                                     |
| 4   | –        | Not used.                                                                   |
| 5   | CONF_FLT | • 1: Hardware or software configuration detected error.  
     |          | • 0: No detected error                                                     |
| 6   | NO_MOD   | • 1: Module is missing or inoperative.  
     |          | • 0: Module is operating.                                                  |
|     |          | **NOTE:** This bit is managed only by modules located in a remote rack with a  
     |          | BME CRA 312 10 adapter module. Modules located in the local rack do not manage this bit that remains at 0. |
| 7   | –        | Not used.                                                                   |
Part V
Quick Start: Example of Counting Module Implementation

Subject of this Part
This part presents an example of implementation of the counting modules.

What Is in This Part?
This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Description of the Application</td>
<td>207</td>
</tr>
<tr>
<td>15</td>
<td>Installing the Application Using Control Expert</td>
<td>209</td>
</tr>
<tr>
<td>16</td>
<td>Starting the Application</td>
<td>231</td>
</tr>
</tbody>
</table>
Chapter 14
Description of the Application

Overview of the Application

At a Glance
The application described in this document is used for sticking labels on boxes. The boxes are carried on a conveyor. A label is stuck onto the box when the latter passes by the two dedicated points. A sensor placed below the conveyor detects any new incoming box. The boxes should arrive at constant intervals. The conveyor motor is fitted with an encoder connected to a counting input module. Any process deflection is monitored and displayed. The application's control resources are based on an operator screen displaying all box positions, the number of labeled boxes and the deflection monitoring.

Illustration
This is the application's final operator screen:
Description of the application

Operating Mode

The operating mode is as follows:

- A **Start** button is used to start the labelling process.
- A **Stop** button interrupts the labelling process.
- When the box arrives at the right time, the **Box on time** indicator lights on.
- In case of process deflection, the box delay time is displayed. If this time has been too long, a **Process deflection** indicator lights on.
Chapter 15
Installing the Application Using Control Expert

Subject of this chapter
This chapter describes the procedure for creating the application described. It shows, in general and in more detail, the steps in creating the different components of the application.

What Is in This Chapter?
This chapter contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1</td>
<td>Presentation of the Solution Used</td>
<td>210</td>
</tr>
<tr>
<td>15.2</td>
<td>Developing the Application</td>
<td>213</td>
</tr>
</tbody>
</table>
Section 15.1
Presentation of the Solution Used

Subject of this section
This section presents the solution used to develop the application. It explains the technological choices and gives the application's creation timeline.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technological Choices Used</td>
<td>211</td>
</tr>
<tr>
<td>Process Using Control Expert</td>
<td>212</td>
</tr>
</tbody>
</table>
**Technological Choices Used**

**At a Glance**

There are several ways of writing a counter application using Control Expert. The one proposed, uses the Modulo Loop Counter Mode available in the BMX EHC 0200 counting input module.

**Technological Choices**

The following table shows the technological choices used for the application.

<table>
<thead>
<tr>
<th>Objects</th>
<th>Choices used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter mode</td>
<td>Use of the Modulo Loop Counter Mode. This mode counts the encoder input pulses. The modulo value is the defined counting limit. When the counting reaches the modulo value, the counter restarts from 0. A positive transition of the capture signal triggers the count value capture in the capture register and the counter restarts from 0. In this application, the modulo value is the constant interval between boxes and the capture signal is sent by the sensor. The module reflex outputs are triggered when the counting exceeds defined thresholds.</td>
</tr>
<tr>
<td>Supervision screen</td>
<td>Use of elements from the library and new objects.</td>
</tr>
</tbody>
</table>
| Main supervision program | This program contains two sections.  
  - The first one, which initiates and uses the Modulo Loop Counter Mode functions, is developed using a Structured Text language (ST).  
  - The Application section, which allows operators screen animation, is created in Ladder Diagram (LD) language. |
Process Using Control Expert

At a Glance

The following logic diagram shows the different steps to follow to create the application. A chronological order must be respected in order to correctly define all of the application elements.

Description

Description of the different types:
Section 15.2
Developing the Application

Subject of this Section
This section gives a step-by-step description of how to create the application using Control Expert.

What Is in This Section?
This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating the Project</td>
<td>214</td>
</tr>
<tr>
<td>Configuration of the Counting Module</td>
<td>215</td>
</tr>
<tr>
<td>Declaration of Variables</td>
<td>218</td>
</tr>
<tr>
<td>Creating the Program for Managing the Counter Module</td>
<td>220</td>
</tr>
<tr>
<td>Creating the Labelling Program in ST</td>
<td>222</td>
</tr>
<tr>
<td>Creating the I/O Event Section in ST</td>
<td>224</td>
</tr>
<tr>
<td>Creating a Program in LD for Application Execution</td>
<td>225</td>
</tr>
<tr>
<td>Creating an Animation Table</td>
<td>228</td>
</tr>
<tr>
<td>Creating the Operator Screen</td>
<td>229</td>
</tr>
</tbody>
</table>
Creating the Project

At a Glance

Developing an application using Control Expert involves creating a project associated with a PLC.

Procedure for Creating a Project

The table below shows the procedure for creating the project using Control Expert.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Launch the Control Expert software.</td>
</tr>
<tr>
<td>2</td>
<td>Click on File then New to select a PLC.</td>
</tr>
<tr>
<td>3</td>
<td>To see all PLC versions, click on the box Show all versions.</td>
</tr>
<tr>
<td>4</td>
<td>Select the processor you wish to use from those proposed.</td>
</tr>
<tr>
<td>5</td>
<td>To create a project with specific values of project settings, check the box <strong>Settings File</strong> and use the browser button to localize the .XSO file (Project Settings file). It is also possible to create a new one. If the <strong>Settings File</strong> box is not checked, default values of project settings are used.</td>
</tr>
<tr>
<td>6</td>
<td>Terminate your configuration, insert a BMX EHC 0200 input module.</td>
</tr>
<tr>
<td>7</td>
<td>Confirm with OK.</td>
</tr>
</tbody>
</table>
Configuration of the Counting Module

At a Glance

Developing a counting application involves choosing the right module and appropriate configuration.

Module Selection

The table below shows the procedure for selecting the counting input module.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Project browser double-click on Configuration then on 0:Bus X and on 0:BMX XBP ••• (Where 0 is the rack number)</td>
</tr>
<tr>
<td>2</td>
<td>In the Bus X window, select a slot (for example slot 1) and double-click</td>
</tr>
<tr>
<td>3</td>
<td>Choose the BMX HEC 0200 counting input module</td>
</tr>
<tr>
<td>4</td>
<td>Confirm with OK.</td>
</tr>
</tbody>
</table>
Counting Module Configuration

The table below shows the procedure for selecting the counting function and configuring the module reflex outputs.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Bus X window, double-click on the BMX ENC 0200 counting input module.</td>
</tr>
<tr>
<td>2</td>
<td>Select a channel (for example Counter 0) and click.</td>
</tr>
<tr>
<td>3</td>
<td>Select the module function Modulo Loop Counter Mode.</td>
</tr>
<tr>
<td>4</td>
<td>In the Config tab, configure the OutputBlock 0 reflex output with a pulse when the counting is greater than the Lower Threshold (Pulse = greater than LT) and the OutputBlock 1 reflex output with a pulse when the counting is greater than the Upper Threshold (Pulse = greater than UT). Then click on the Event value and select Enable.</td>
</tr>
<tr>
<td>5</td>
<td>Click on the Adjust tab and enter the modulo value (for example 50).</td>
</tr>
</tbody>
</table>
**Declaration of I/O objects**

The table below shows the procedure for declaring the I/O Derived Variable.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the BMX EHC 0200 window, click on the BMX EHC 0200 and then on the I/O objects tab.</td>
</tr>
<tr>
<td>2</td>
<td>Click on the I/O object prefix address %CH then on the Update grid button, the channel address appears in the I/O object grid.</td>
</tr>
<tr>
<td>3</td>
<td>Click on the line %CHO.1.0 and then enter a channel name in the Prefix for name zone.</td>
</tr>
<tr>
<td>4</td>
<td>Now click on different Implicit I/O object prefix addresses then Update grid button to see the names and addresses of the implicit I/O objects.</td>
</tr>
</tbody>
</table>
Declaration of Variables

At a Glance

All of the variables used in the different sections of the program must be declared.

Undeclared variables cannot be used in the program.

NOTE: For more information, refer to chapter Data Editor (see EcoStruxure™ Control Expert, Operating Modes).

Procedure for Declaring Variables

The table below shows the procedure for declaring application variables.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In Project browser / Variables &amp; FB instances, double-click on Elementary variables</td>
</tr>
<tr>
<td>2</td>
<td>In the Data editor window, select the box in the Name column and enter a name for your first variable.</td>
</tr>
<tr>
<td>3</td>
<td>Now select a Type for this variable.</td>
</tr>
<tr>
<td>4</td>
<td>When all your variables are declared, you can close the window.</td>
</tr>
</tbody>
</table>

Variables Used for the Application

The following table shows the details of the variables used in the application.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>EBOOL</td>
<td>Startup request for the labelling process.</td>
</tr>
<tr>
<td>Stop</td>
<td>EBOOL</td>
<td>Stop the labelling process.</td>
</tr>
<tr>
<td>Last_Box_late</td>
<td>BOOL</td>
<td>The process is in deflection.</td>
</tr>
<tr>
<td>Nb_Box</td>
<td>DINT</td>
<td>Number of labelled boxes.</td>
</tr>
<tr>
<td>Position_0</td>
<td>BOOL</td>
<td>Box at the beginning of the conveyor.</td>
</tr>
<tr>
<td>Position_1</td>
<td>BOOL</td>
<td>Box with the first label.</td>
</tr>
<tr>
<td>Position_2</td>
<td>BOOL</td>
<td>Box with the two labels.</td>
</tr>
<tr>
<td>First_Labelling_Point</td>
<td>DINT</td>
<td>Lower Threshold value.</td>
</tr>
<tr>
<td>Second_Labelling_Point</td>
<td>DINT</td>
<td>Upper Threshold value.</td>
</tr>
<tr>
<td>Deflection_Parameter</td>
<td>DINT</td>
<td>Deflection alarm triggering value.</td>
</tr>
<tr>
<td>Waiting_First_Part</td>
<td>BOOL</td>
<td>The first box is waited.</td>
</tr>
<tr>
<td>Waiting_Other_Parts</td>
<td>BOOL</td>
<td>The first box has already passed.</td>
</tr>
</tbody>
</table>
The following screen shows the application variables created using the data editor:

![Data Editor Screen](image)

### Data Editor

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition_Parameter</td>
<td>DINT</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Encoder</td>
<td>T_UNSG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First_Labelling_Point</td>
<td>DINT</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Last_Box_Labe</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No_Box</td>
<td>DINT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Position_0</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position_1</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position_2</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Second_Labelling_Point</td>
<td>DINT</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Shop</td>
<td>EBOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waiting_First_Past</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waiting_Other_Parts</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:** Click on the icon in front of the derived variable **Encoder** to expand the I/O objects list.
Creating the Program for Managing the Counter Module

At a Glance

Two sections are declared in the MAST task:

- The Labelling Program section (See Creating the Labelling Program in ST, page 222), written in ST, initiates and uses the Modulo Loop Counter Mode functions and I/O objects,
- The Application section (See Creating a Program in LD for Application Execution, page 225), written in LD, executes the counting start-up and the operator screen animation.

Process Chart

The following screen shows the process chart.

```
Function Enabling

Threshold definitions

Process deflection

False

Deflection Alarm ON

True

Deflection Alarm OFF
```

Description of the Labelling Program Section

The following table describes the different steps of the process chart.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions enabling</td>
<td>Enables the Modulo Mode functions used in the application.</td>
</tr>
<tr>
<td>Threshold definitions</td>
<td>The values of the thresholds, on which depend the reflex outputs, are defined in this step.</td>
</tr>
<tr>
<td>Process deflection</td>
<td>Test if the capture value is greater than the deflection parameter</td>
</tr>
<tr>
<td>Deflection Alarm ON</td>
<td>If the result of the process deflection test is true, the alarm is ON.</td>
</tr>
<tr>
<td>Deflection Alarm OFF</td>
<td>If the result of the process deflection test is false, the alarm is OFF.</td>
</tr>
</tbody>
</table>
Creating the Labelling Program in ST

At a Glance
This section initiates and uses the Modulo Loop Counter Mode functions and objects.

Illustration of the Labelling Program Section
This section below is part of the MAST task. It has no condition defined for it so it is permanently executed:

(*Functions Enabling*)
(*Authorizes Input SYNC to synchronize and start the counting function*)
Encoder.VALID_SYNC:=Waiting_First_Part;
IF Waiting_First_Part
  THEN nb_box := 0;
END IF;
(*Once the first part has passed below the sensor, the other functions are enabled.*)
IF Waiting_Other_Parts
  THEN
(*Authorizes captures into the Capture 0 register*)
  Encoder.VALID_CAPT_0:=1;
(*Authorizes comparators to produce its results*)
  Encoder.COMPARE_ENABLE:=1;
(*Call Event task when Counter Roll over*)
  Encoder.EVT_MODULO_ENABLE:=1;
(*Enable the output block functions*)
  Encoder.OUTPUT_BLOCK_0_ENABLE:=1;
  Encoder.OUTPUT_BLOCK_1_ENABLE:=1;
ELSE
(*Function disabling when the conveyor is stopped*)
  Encoder.VALID_CAPT_0:=0
  Encoder.COMPARE_ENABLE:=0
  Encoder.EVT_MODULO_ENABLE:=0
  Encoder.OUTPUT_BLOCK_0_ENABLE:=0
  Encoder.OUTPUT_BLOCK_1_ENABLE:=0
END IF

(*Definition of the lower and upper threshold values*)
Encoder.LOWER_TH_VALUE:=First_Labelling_Point;
Encoder.UPPER_TH_VALUE:=Second_Labelling_Point;

(*Process Deflection Watching*)
IF Encoder.CAPT_0_VALUE>deflection_parameter=true
   THEN last_box_late:=1; (*Default light set ON*)
   ELSE last_box_late:=0; (*Default light set OFF*)
END IF

(*If the next part arrives just in the right time, the green indicator lights on*)
IF Encoder.CAPT_0_VALUE = 0
   THEN Last_Box_On_Target :=1 (*Green light set ON*)
   ELSE Last_Box_On_Target :=0 (*Green light set OFF*)
END IF

Procedure for Creating an ST Section

The table below shows the procedure for creating an ST section for the application.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In Project Browser\Program\Tasks, double-click on MAST,</td>
</tr>
<tr>
<td>2</td>
<td>Right-click on Section then select New section. Give your section a name and select ST language.</td>
</tr>
<tr>
<td>3</td>
<td>The name of your section appears and can now be edited by double-clicking on it.</td>
</tr>
</tbody>
</table>
| 4 | To use the I/O object, right-click in the editor then click on Data selection and on .
   Click on on the front of the I/O derived variable Encoder and the list of the I/O objects appears.
   Click on the one you need and confirm with OK. |

NOTE: In the Data selection windows, the IODDT checkbox must be checked to have access to the I/O derived variable Encoder.
Creating the I/O Event Section in ST

At a Glance

This section is called when the modulo value is reached.

Illustration of the Event Section

The section below is part of the Event task:

(*Number of labelled boxes is incremented at the Modulo Event *)

INC(Nb_Box);

Procedure for Creating an ST Section

The table below shows the procedure for creating an I/O Event.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In Project Browser\Program, double-click on Events</td>
</tr>
<tr>
<td>2</td>
<td>Right click on I/O Events then select New Event section. Give your section a number, for this example select 0, and then select ST language</td>
</tr>
<tr>
<td>3</td>
<td>Confirm with OK and the edition window appears.</td>
</tr>
</tbody>
</table>
Creating a Program in LD for Application Execution

At a Glance

This section executes the counting start up and the operator screen animation.
Illustration of the Application Section

The section below is part of the MAST task:

```
FB1_1

<table>
<thead>
<tr>
<th>RS</th>
<th>EN</th>
<th>ENo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Makes the counter enabled
Encoder FORCE_ENABLED

Numeral current value is less than the Lower Threshold
Encoder Counter_Low

Position_0

Numeral current value is within the Thresholds
Encoder Counter_Mid

Position_1

Numeral current value is greater than the Upper Thresholds
Encoder Counter_High

Position_2

FB2_2

<table>
<thead>
<tr>
<th>RS</th>
<th>EN</th>
<th>ENo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>p</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

First part waiting
Encoder Waiting_First_Part

FB3_3

<table>
<thead>
<tr>
<th>RS</th>
<th>EN</th>
<th>ENo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Waiting_Other_Parts
Encoder Waiting_Other_Parts
```
Description of the Application Section

- The first line is used to command the counter.
- The other three lines are used to simulate the different box positions on the conveyor.
- The last part is used to control the variables which allow the function enabling (See Illustration of the Labelling Program Section, page 222).
- When Run switches to '1', Waiting_First_Part is set to '1'.
- A sensor signal triggers the flag Sync_ref_flag which resets Waiting_first_part to '0' and sets Waiting_other_parts to '1'.

Procedure for Creating an LD Section

The table below describes the procedure for creating part of the Application section.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In Project Browser\Program\Tasks, double-click on MAST.</td>
</tr>
<tr>
<td>2</td>
<td>Right click on Section then select New section. Name this section Application, then select the language type LD. The Edit window opens.</td>
</tr>
<tr>
<td>3</td>
<td>To create the contact Encoder.Sync_Ref_Flag, click on ( \text{Encoder.Sync_Ref_Flag} ) then place it in the editor. Double-click on this contact then on ( \text{Encoder.Sync_Ref_Flag} ). The Instance Selection window opens. Validate the Inside structure checkbox and click on ( ) in front of the Encoder variable and select Sync_Ref_Flag in the list. Confirm with OK.</td>
</tr>
<tr>
<td>4</td>
<td>To use the RS block you must instantiate it. Right click in the editor then click on Select data and on ( \text{Encoder.Sync_Ref_Flag} ). Click on the Function and Function Block Types tab. Click on Libset and select the RS block in the list then confirm with OK and position your block. To link the Encoder.Sync_Ref_Flag contact to the S Rinput of the RS block, align the contact and the input horizontally, click on ( ) and position the link between the contact and the input.</td>
</tr>
</tbody>
</table>

NOTE: For more information on creating an LD section, refer to chapter LD Editor.
Creating an Animation Table

At a glance
An animation table is used to monitor the values of variables, and modify and/or force these values. Only those variables declared in Variables & FB instances can be added to the animation table.

**NOTE:** For more information, refer to chapter Animation Tables (see EcoStruxure™ Control Expert, Operating Modes).

Procedure for Creating an Animation Table

The table below shows the procedure for creating an animation table.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Project browser, right click on Animation tables. The edit window opens.</td>
</tr>
<tr>
<td>2</td>
<td>Click on first cell in the Name column, then on the button, and add the variables you require.</td>
</tr>
</tbody>
</table>

Animation Table Created for the Application

The following screen shows the animation table used by the application:

![Animation Table](image)

**NOTE:** The animation table is dynamic only in online mode (display of variable values)
Creating the Operator Screen

At a Glance

The operator screen is used to animate graphic objects that symbolize the application. These objects can belong to the Control Expert library, or can be created using the graphic editor.

**NOTE:** For more information, refer to chapter *Operator screens* (see *EcoStruxure™ Control Expert, Operating Modes*).

Illustration on an Operator Screen

The following illustration shows the application operator screen:

![Operator Screen Illustration](image_url)

**NOTE:** To animate objects in online mode, you must click on ![Animate](image_url). By clicking on this button, you can validate what is written.
Procedure for Creating an Operator Screen

The table below shows the procedure for creating the Start button.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Project browser, right click on Operator screens and click on New screen. The operator screen editor appears.</td>
</tr>
<tr>
<td>2</td>
<td>Click on the and position the new button on the operator screen. Double click on the button and in the Control tab, select the Run variable by clicking the button and confirm with OK. Then, enter the button name in the text zone.</td>
</tr>
</tbody>
</table>

The table below shows the procedure for inserting and animating the conveyor.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the Tools menu, select Operator screens Library. Double click on Machine then Conveyor. Select the dynamic conveyor from the runtime screen and Copy (Ctrl+C) then Paste (Ctrl+V) it into the drawing in the operator screen editor.</td>
</tr>
<tr>
<td>2</td>
<td>The conveyor is now in your operator screen. You now need a variable to animate the wheels. Select your conveyor then click on . A line on the wheel is selected. Press enter and the object properties window opens. Select the Animation tab and enter the concerned variable, by clicking on (in the place of %MW0). In our application, this will be Encoder._INPUT_A, the physical input A state. Confirm with Apply and OK.</td>
</tr>
<tr>
<td>3</td>
<td>Click on to select the other lines one by one and apply the same procedure.</td>
</tr>
</tbody>
</table>

NOTE: In the Instance Selection, tick the IODDT checkbox and click on to access the I/O objects list.

The table below shows the procedure for inserting and animating a display.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Click on and position it on the operator screen. Double click on the text and select the Animation tab.</td>
</tr>
<tr>
<td>2</td>
<td>Tick the Animated Object checkbox, select the concerned variable by clicking on and confirm with OK.</td>
</tr>
</tbody>
</table>
Chapter 16
Starting the Application

Execution of Application in Standard Mode

At a Glance
Standard mode working requires the use of a PLC and a BMX EHC 0200 with an encoder and a sensor linked to its inputs.

Outputs wiring
The actuators are connected as follow:

![Diagram showing connections between BMX EHC 0200 and actuators]
The assignment of the 10 pins connector is as follow:

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24V_IN</td>
<td>24 VDC input for input supply</td>
</tr>
<tr>
<td>2</td>
<td>GND_IN</td>
<td>0 VDC input for input supply</td>
</tr>
<tr>
<td>5</td>
<td>Q0-1</td>
<td>Q0 output for counting channel 1</td>
</tr>
<tr>
<td>6</td>
<td>Q0-0</td>
<td>Q0 output for counting channel 0</td>
</tr>
<tr>
<td>7</td>
<td>Q1-1</td>
<td>Q1 output for counting channel 1</td>
</tr>
<tr>
<td>8</td>
<td>Q1-0</td>
<td>Q1 output for counting channel 0</td>
</tr>
<tr>
<td>9</td>
<td>24V_OUT</td>
<td>24 VDC input for output supply</td>
</tr>
<tr>
<td>10</td>
<td>GND_OUT</td>
<td>0 VDC input for output supply</td>
</tr>
</tbody>
</table>
Inputs Wiring

The encoder and the sensor are connected as follows:

The assignment of the 16 pins connector is as follows:
Starting the Application

Description:

<table>
<thead>
<tr>
<th>Pin number</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 7, 8</td>
<td>24V_SEN</td>
<td>24 VDC output for sensor supply</td>
</tr>
<tr>
<td>5, 6, 13, 14</td>
<td>GND_SEN</td>
<td>0 VDC output for sensor supply</td>
</tr>
<tr>
<td>15, 16</td>
<td>FE</td>
<td>Functional ground</td>
</tr>
<tr>
<td>3</td>
<td>IN_A</td>
<td>Input A</td>
</tr>
<tr>
<td>4</td>
<td>IN_SYNC</td>
<td>Synchronization input</td>
</tr>
<tr>
<td>9</td>
<td>IN_B</td>
<td>Input B</td>
</tr>
<tr>
<td>10</td>
<td>IN_EN</td>
<td>Enable input selected</td>
</tr>
<tr>
<td>11</td>
<td>IN_REF</td>
<td>Homing input</td>
</tr>
<tr>
<td>12</td>
<td>IN_CAP</td>
<td>Capture input</td>
</tr>
</tbody>
</table>

Application Execution

The table below shows the procedure for launching the application in standard mode:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In the PLC menu, click on Standard Mode,</td>
</tr>
<tr>
<td>2</td>
<td>In the Build menu, click on Rebuild All Project. Your project is generated and is ready to be transferred to the PLC. When you generate the project, you will see a results window. If there is an error in the program, Control Expert indicates its location if you click on the highlighted sequence.</td>
</tr>
<tr>
<td>3</td>
<td>In the PLC menu, click on Connection. You are now connected to the PLC.</td>
</tr>
<tr>
<td>4</td>
<td>In the PLC menu, click on Transfer project to PLC. The Transfer project to PLC window opens. Click on Transfer. The application is transferred to the PLC.</td>
</tr>
<tr>
<td>5</td>
<td>In the PLC, click on Execute. The Execute window opens. Click on OK. The application is now being executed (in RUN mode) on the PLC.</td>
</tr>
</tbody>
</table>
Index

B
BMXEH0200, 22
BMXXSP0400, 50
BMXXSP0600, 50
BMXXSP0800, 50
BMXXSP1200, 50

C
certifications, 36
channel data structure for all modules
   T_GEN_MOD, 192, 192
channel data structure for counting modules
   T_SIGNED_CPT_BMX, 185, 190
   T_UNSIGNED_CPT_BMX, 185, 190
configuring, 119
Counting Events, 79

D
debugging, 151
diagnosing, 67

E
event counting, 84

F
filtering, 58
free large counter, 99
frequency mode, 83
functions, 56

G
grounding accessories, 50
   BMXXSP0400, 50
   BMXXSP0600, 50
   BMXXSP0800, 50
   BMXXSP1200, 50
   STBXSP3010, 50
   STBXSP3020, 50

I
input interface blocks, 57
installing, 27, 109

M
MOD_FLT, 203
modulo loop counter, 95

O
one shot counter, 92

P
parameter settings, 173
period measuring, 86
pulse width modulation, 107

Q
quick start, 205

R
ratio, 89

S
settings, 143
standards, 36
STBXSP3010, 50
STBXSP3020, 50

T
T_GEN_MOD, 192, 192
T_M_CPT_STD_IN_2, 195
T_M_CPT_STD_IN_8, 195
T_SIGNED BMX, 185
T_SIGNED_CPT_BMX, 190
T_UNSIGNED_CPT_BMX, 185, 190
terminal blocks
   connecting, 27
   installing, 27

W
wiring accessories, 27