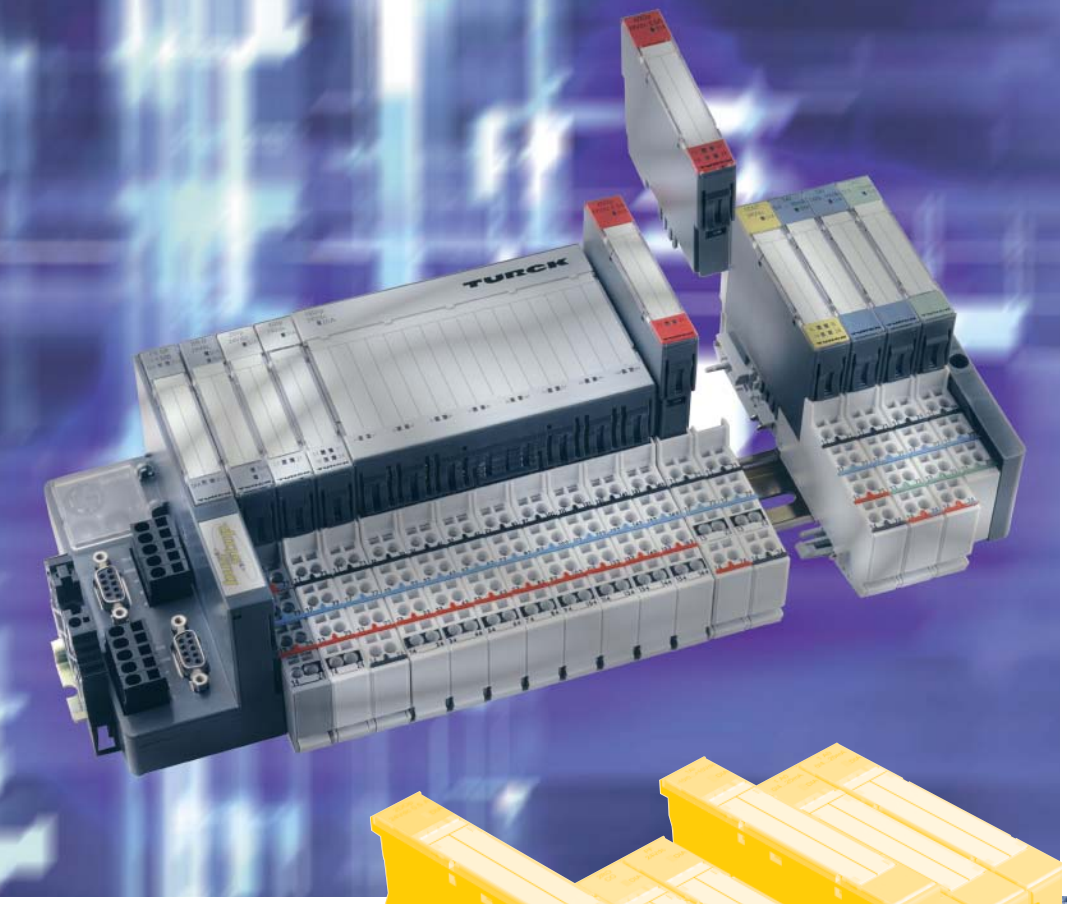
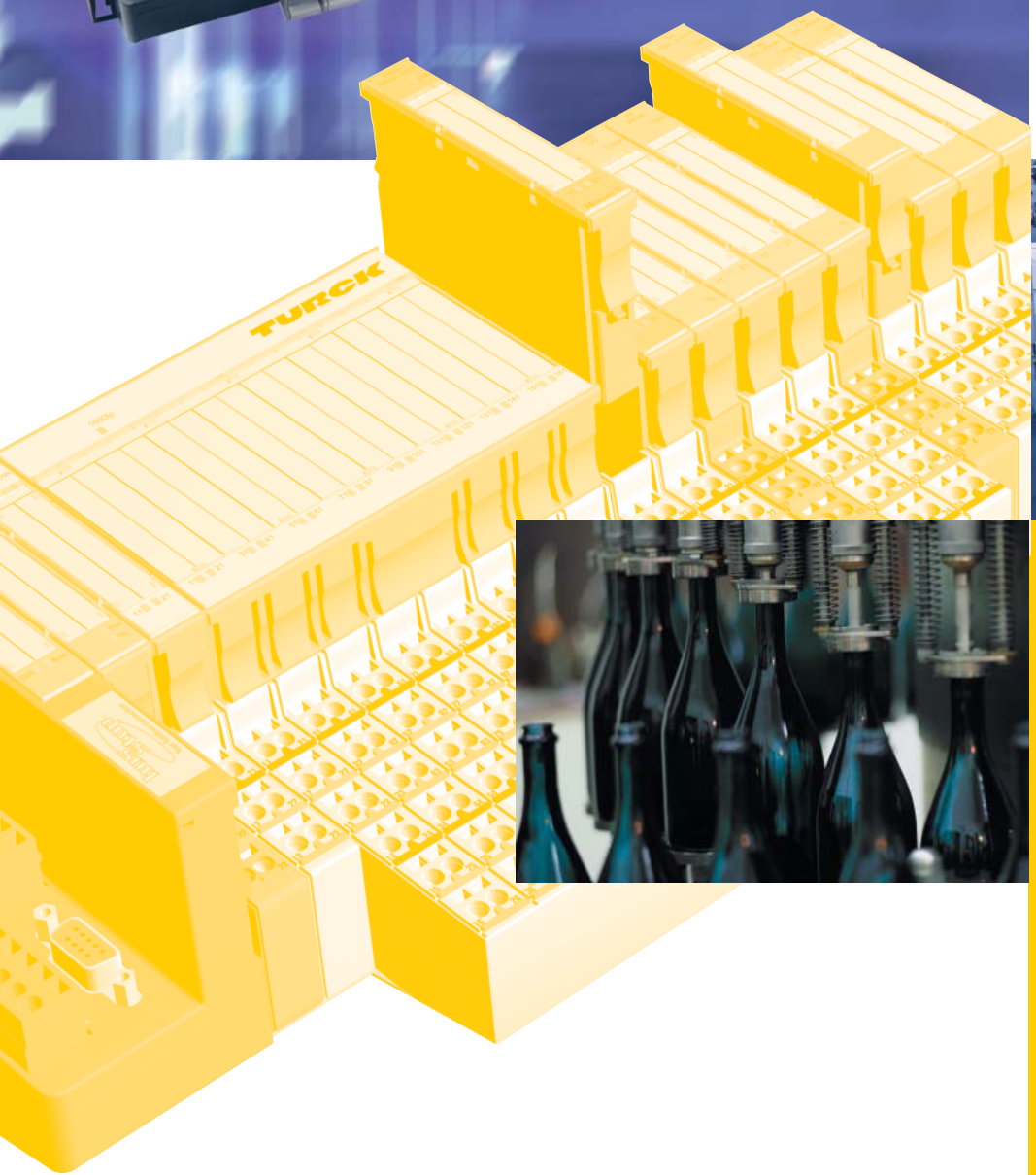
TURCK

Industrial Automation

# BL20 – I/O-MODULES

## BL20-E-2CNT-2PWM





Edition 08/2012

## Warning!
## Before commencing the installation

- Disconnect the power supply of the device.
- Ensure that devices cannot be accidentally restarted.
- Verify isolation from the supply.
- Earth and short circuit.
- Cover or enclose neighboring units that are live.
- Follow the engineering instructions of the device concerned.
- Only suitably qualified personnel in accordance with EN 50 110-1/-2 (VDE 0 105 Part 100) may work on this device/system.
- Before installation and before touching the device ensure that you are free of electrostatic charge.
- The functional earth (FE) must be connected to the protective earth (PE) or to the potential equalization. The system installer is responsible for implementing this connection.
- Connecting cables and signal lines should be installed so that inductive or capacitive interference do not impair the automation functions.
- Install automation devices and related operating elements in such a way that they are well protected against unintentional operation.
- Suitable safety hardware and software measures should be implemented for the I/O interface so that a line or wire breakage on the signal side does not result in undefined states in the automation devices.
- Ensure a reliable electrical isolation of the low voltage for the 24 volt supply. Only use power supply units complying with IEC 60 364-4-41 (VDE 0 100 Part 410) or HD 384.4.41 S2.
- Deviations of the mains voltage from the rated value must not exceed the tolerance limits given in the specifications, otherwise this may cause malfunction and dangerous operation.
- Emergency stop devices complying with IEC/EN 60 204-1 must be effective in all operating modes of the automation devices. Unlatching the emergency-stop devices must not cause restart.
- Devices that are designed for mounting in housings or control cabinets must only be operated and controlled after they have been installed with the housing closed. Desktop or portable units must only be operated and controlled in enclosed housings.
- Measures should be taken to ensure the proper restart of programs interrupted after a voltage dip or failure. This should not cause dangerous operating states even for a short time. If necessary, emergency-stop devices should be implemented.
- Wherever faults in the automation system may cause damage to persons or property, external measures must be implemented to ensure a safe operating state in the event of a fault or malfunction (for example, by means of separate limit switches, mechanical interlocks etc.).
- The electrical installation must be carried out in accordance with the relevant regulations (e. g. with regard to cable cross sections, fuses, PE).
- All work relating to transport, installation, commissioning and maintenance must only be carried out by qualified personnel. (IEC 60 364 and HD 384 and national work safety regulations).
- All shrouds and doors must be kept closed during operation.

# Table of contents

# 1 About this manual

## 1.1 General

This manual describes the technology module BL20-E-2CNT-2PWM which is part of the product line BL20.

In addition to the description of the technical features und functions, it also contains a description of the I/O module's representation in different fieldbus systems.

### 1.1.1 Additional documentation

- D300717 "BL20 I/O modules - Hardware and Engineering"
- D300956 "BL20/BL67– USER MANUAL FOR PROFIBUS-DPV1"
- D301106 "BL20 – ECO GATEWAY FOR PROFIBUS-DP"
- D301108 "BL20 – ECO GATEWAY FOR CANOPEN"
- D301087 „BL20 –ANWENDERHANDBUCH FÜR CANOPEN“

**Note**

All manuals concerning the product line BL20 can be found on our homepage.

## 1.2 Description of symbols used

**Warning**

This sign can be found next to all notes that indicate a source of hazards. This can refer to danger to personnel or damage to the system (hardware and software) and to the facility.

This sign means for the operator: work with extreme caution.

**Attention**

This sign can be found next to all notes that indicate a potential source of hazards.

This can refer to possible danger to personnel and damages to the system (hardware and software) and to the facility.

**Note**

This sign can be found next to all general notes that supply important information about one or more operating steps.

These specific notes are intended to make operation easier and avoid unnecessary work due to incorrect operation.

## 1.3 Overview

**Attention**

Please read this section carefully. Safety aspects cannot be left to chance when dealing with electrical equipment.

This manual includes all information necessary for the prescribed use of TURCK devices. It has been specially conceived for personnel with the necessary qualifications.

### 1.3.1 Prescribed use

Appropriate transport, storage, deployment and mounting as well as careful operating and thorough maintenance guarantee the trouble-free and safe operation of these devices.

**Warning**

The devices described in this manual must be used only in applications prescribed in this manual or in the respective technical descriptions, and only with certified components and devices from third party manufacturers.

### 1.3.2 Notes concerning planning /installation of this product

**Warning**

All respective safety measures and accident protection guidelines must be considered carefully and without exception.

## 1.4 List of revisions

In comparison to the previous manual edition, the following changes/ revisions have been made:

*Table 1-1: List of revisions*

| Chapter | Subject/ Description | new | changed |
|---|---|---|---|
| Chap. 12 | Representation of the BL20-E-2CNT-2PWM in EtherNet/IP™ | X | |

**Note**

The publication of this manual renders all previous editions invalid.

# 2 Getting Started - first steps for operating this module

## 2.1 General information about the module

The BL20-E-2CNT-2PWM provides 2 counter channels (CNT1 and CNT 2) with 3 count inputs each DI0 (A), DI1 (B) and DI2 (Z) as well as 2 channels with 2 outputs each, P0 and P1 (frequency output) and D1 and D2 (direction).

### 2.1.1 Counter inputs

The module's counter inputs allow the connection of encoders as well as of count- and direction signals, with the direction evaluation being optional.

**operation modes**

- Counting
  - pulse and direction, → see page 4-10
  - AB mode, → see page 4-11
- Measurement
  - Frequency measurement / rotational speed measurement, → see page 4-19
  - Period duration measurement, → see page 4-20

### 2.1.2 PWM outputs

Each channel provides two PWM outputs P1/ D1 and P2/ D2.

P1 and P2 are used for frequency output.

The logical status of the outputs D1 and D2 can be used to define the rotation direction. The outputs can also be used independently of the PWM.

The outputs Px serve to give out a square wave signal with a defined mark-to-space ratio, a defined period duration and a defined number of pulses.

**operation modes**

- Period duration /duty cycle, → see page 5-3
- High time / low time definition, → see page 5-5

## 2.2 Getting Started

### 2.2.1 Count function

The following section describes the general procedure to read the count value of counter **CNT1** of BL20-E-2CNT-2PWM.

**Prerequisites/ start conditions**

The default settings of the module parameters allow immediate access to the module's count value.

- Parameterization:
  mode CNT1 = 0000 = pulse/direction, single sample (Parameter data of the module (page 3-10)).
  The signals on input A1 are counted, the count direction is defined via signal B1.
  Z1 can be used as a HW gate (→ see below).
- Mapping of the count value into the process data:
  The count value is mapped into the process data (Process input/ check-back interface (page 3-14)):
  ADR AUX REG1 RD DATA = 0×20
  → 0×20 = register no. of REG_CNT1_CNT (actual count value of CNT1) → see also Register interface (page 9-4).
  → The count value is mapped into bytes 12 to 15 of the process data (→ see also Process input/ check-back interface (page 3-14)).

**Current count value**

Process input/ check-back interface (page 3-14), byte 12 to byte 15:

AUX_REG1_RD_DATA, Byte 0,

to
AUX_REG1_RD_DATA, Byte 3 contain the current count value of **CNT1**

**Enable**

The counter function unit of the CNT-inputs is generally enabled per **default setting** .
CNT1_ GENERAL_ DISABLE = 0
(→ see Process output/ control interface byte 0, bit 0, page 3-19).

The function unit can be generally disabled by setting
CNT1_ GENERAL_ DISABLE = 1

The general enabling of the counter function unit is displayed via
STS_CNT1_ GENERAL_ EN = 1
(→ see Process input/ check-back interface, Byte 0, Bit 0, page 3-14).

**1** If the counter has to count from a defined start value, this value has to be written into the load value register REG_CNT1_LOADVAL (→ see Internal registers - reading and writing (page 9-2)).

**2** A state change from 0 → 1 in control bit CNT1_ SW_LR of the process output/ control interface executes a Latch Retrigger which is necessary to transfer the load value.

**3** An executed Latch Retrigger is confirmed in the process input/ check-back interface with MSG_CNT1_SW_LR= 1.

**4** The load value (REG_CNTx_LOADVAL) is transferred into the register for the current count value CNT1 REG_CNTx_CNT (→ see also Register interface (page 9-4)).

**5** If the start conditions for the count operation are defined, the count operation has to be enabled using either the HW **or** the SW gate.

**5.1** The default parameterization allows an immediate enabling of the count function via a signal at input Z (HW gate).
Parameters:
Mode Z1 = 0001 = HW gate CNT (→ see Parameter data of the module (page 3-10))
**or**

**5.2** If the SW gate is to be used to enable the counting, the user has to set the process output bit CNT1_ ENABLE 0 → 1
(→ see Process output/ control interface (page 3-19), Byte 0, Bit 1, ).

**6** The count operation is enabled with CNT1_ ENABLE = 1.

**7** The pulses are counted according to the parameterized operation mode.
(→ siehe Process input/ check-back interface (page 3-14), byte 0, bit 1, STS_CNT1_ RUN = 1)

**8** In default parameterization, the data (→ see above Prerequisites/ start conditions (page 2-3)) can be read out from registers AUX_REG_RD_DATA, byte 0 to AUX_REG_RD_DATA, byte 3 (bytes 12 to 15 of the process input data /check-back interface ((→ see also Current count value (page 2-3)).

**Note**

For further functions of the BL20-E-2CNT-2PWM (parameterization, process image, internal registers etc.), please read the following chapters.

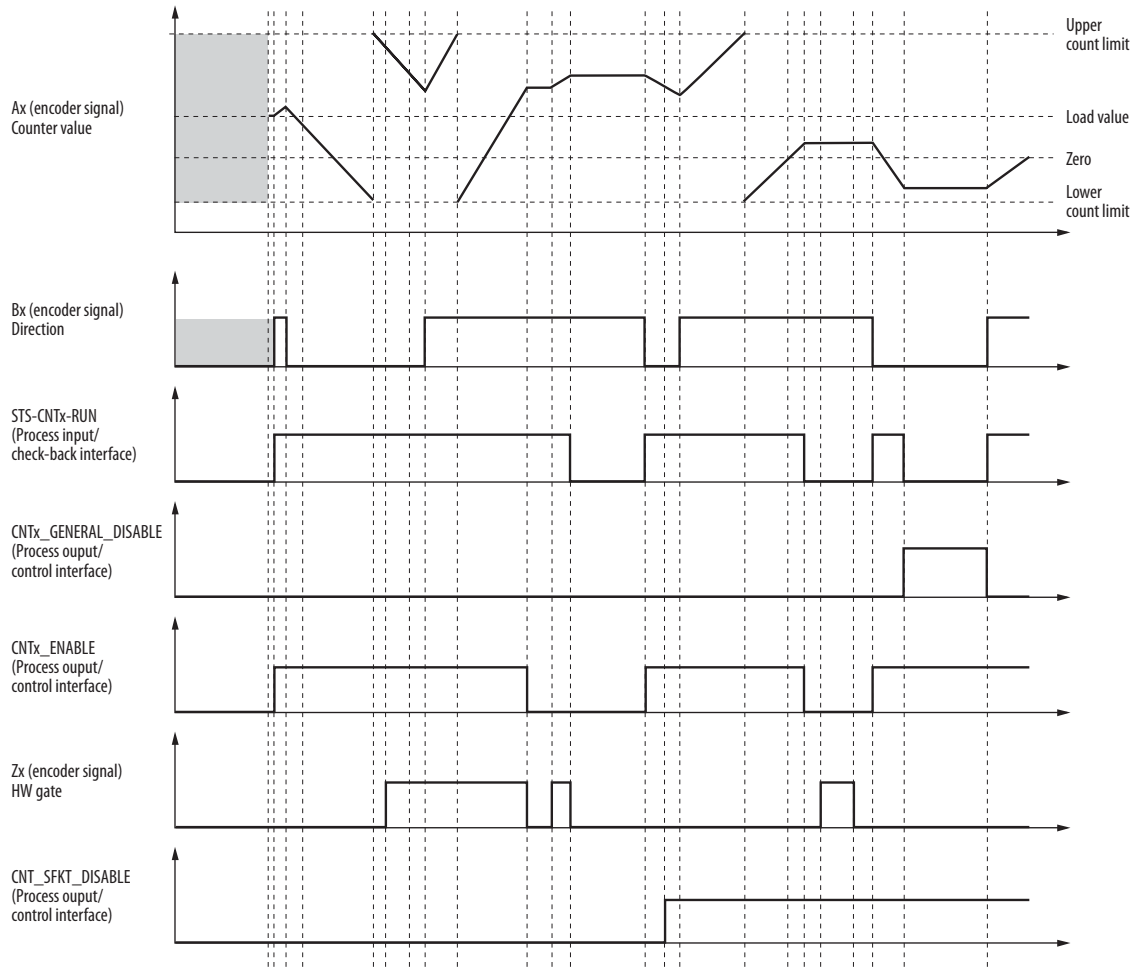



*Figure 2-1: Count function, simplified representation*

## 2.2.2 PWM function

The following section describes the current procedure to output pulses at **PWM1** of BL20-E-2CNT-2PWM:

**Prerequisites/ start conditions**

The default parameterization fulfils all necessary prerequisites/ start conditions for a pulse output:

- Operation mode: Period duration/ duty cycle definition, → see Parameter data of the module (page 3-10) and Period Duration / Duty Cycle Definition (page 5-3)
- Period duration: AB mode, → see REG_PWM1_PD (page 9-9)
- Duty cycle: AB mode, → see REG_PWM1_DC (page 9-9)
- PWM1_ SINGLE = 0 (continuous enable, → see Process output/ control interface (page 3-19)

**Note**

If a defined number of pulse is to be output, the bit PWM1_ SINGLE has to be set = 1 (Process output/ control interface (page 3-19)) **and** the number of the pulses to be output has to be defined via REG_PWM1_CNTSV (load value register of PWM1, register nr. 0 × 64, → see chapter 9, Register interface (page 9-4)).

**Enable**

The pulse output **is generally enabled via the default settings**.
PWM1_ GENERAL_ DISABLE = 0
(→ see Process output/ control interface Byte 2, Bit 0, page 3-19).
The pulse output can be generally disabled using PWM1_ GENERAL_ DISABLE = 1.

An enabled PWM function is displayed via
STS_PWM1_GENERAL_ EN = 1
(→ see Process input/ check-back interface, Byte 4, Bit 4, page 3-14).

**1** The user has to enable the pulse output using either the HW or the SW gate.

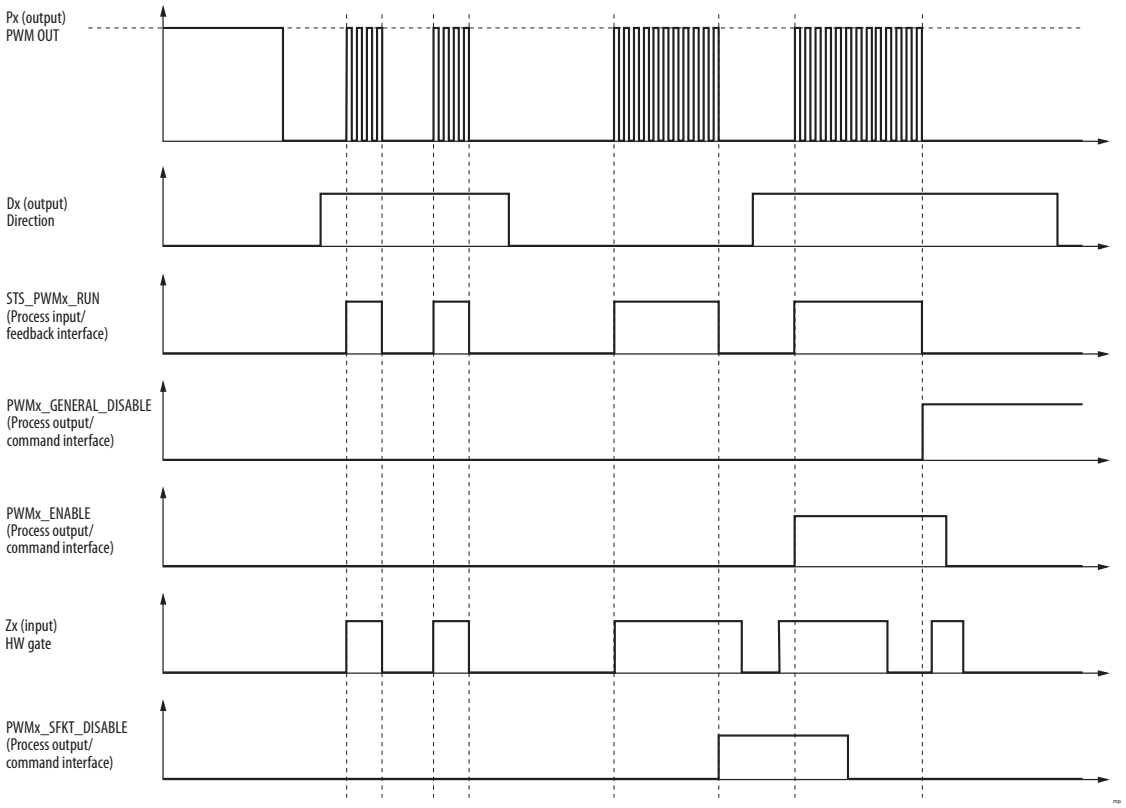
**1.1** Hardware gate:
Input „Z“ can be parameterized as HW gate for the pulse output:
Parameters:
Mode Z1 = 1000 = HW gate CNT (→ ® see Parameter data of the module (page 3-10))
**or**

**1.2** Software gate:
The software gate is set via the process output bit
PWM1_ ENABLE= 1.
(Process output/ control interface byte 2, bit 1, page 3-19).

**2** The pulse output is enabled.

**3** The pulses are output according to the parameterized period duration and the parameterized duty cycle (see above).
((→ see Process input/ check-back interface (page 3-14), byte 4, bit 5, STS_PWM1_RUN = 1)

*Figure 2-2: PWM function, simplified representation*




**Note**

For further functions of the BL20-E-2CNT-2PWM (parameterization, process image, internal registers etc.), please read the following chapters.

# 3 General description of the module

## 3.1 General information about the register interface

The module BL20-E-2CNT-2PWM contains an internal communication interface, the Register interface → see chapter 9).

The data area of the register interface is organized in double words and consists of 128 registers large.

The 128 registers of the register interface allow access to all important information, data and settings concerning the module:

- Module information (hardware-version, software-version, etc.)
- process data
- parameter data
- Diagnostic data

**Note**

Please find a detailed description of the register interface in chapter 9, Register interface.

Example for registers in the register interface:

*Table 3-1: Example for registers in the register interface*

| Register name | No. | Meaning |
|---|---|---|
| REG_HW_VER | 0×02 | hardware-version |
| REG_CONFIG_ERRSTS | 0×0A | report of configuration errors |
| REG_DATA_IN1 byte 3-0 | 0×0C | process input data 1 |
| REG_PARA1 byte 3-0 | 0×1C | parameter data 1 |
| REG_CNT1_LOADVAL | 0×23 | load value CNT1 |

**Note**

A detailed description of the procedure for the read and the write process can be found in chapter 9, Internal registers - reading and writing (page 9-2).

### 3.1.1 Mapping register contents into process data

A subset of the data in the register interface is mapped into the Process data of the module (page 3-14) in order to allow direct external access.

Bytes 8 -23 of the process data allow reading and writing 4 32-bit-registers of the register interface and can be allocated as required.

**Addressing registers to be mapped**

The address assignment of the register contents to be mapped can be carried out either via the process data or via the module parameters:

1. Address assignment via process data
   The address for **one** register access is directly defined via the Process output/ control interface (page 3-19), byte 6 and 7 (REG_WR_ADR and REG_RD_ADR).
2. Address assignment via parameters
   The address of **three** further registers to be mapped may be definedvia Parameter data of the module (page 3-10), Byte 10 to Byte 15 (ADR_AUX_REG1_RD_DATA to ADR_AUX_REG3_WR_DATA).
   Default-mapping per parameter setting

*Table 3-2: Default-mapping per parameter setting*

| Parameters | Default-Parameterization<br>Register no. | Access | Register content | |
|---|---|---|---|---|
| ADR AUX REG1 RD DATA | 0x60 | RD | REG_PWM1_PD | period duration PWM1 |
| ADR AUX REG2 RD DATA | 0x61 | RD | REG_PWM1_DC | mark-to-space ratio PWM1 |
| ADR AUX REG3 RD DATA | 0x70 | RD | REG_PWM2_PD | period duration PWM2 |
| ADR AUX REG1 WR DATA | 0x20 | WR | REG_CNT1_CNT | current value CNT1 |
| ADR AUX REG2 WR DATA | 0x21 | WR | REG_CNT1_MV | measured value CNT1 |
| ADR AUX REG3 WR DATA | 0x40 | WR | REG_CNT2_CNT | current value CNT2 |

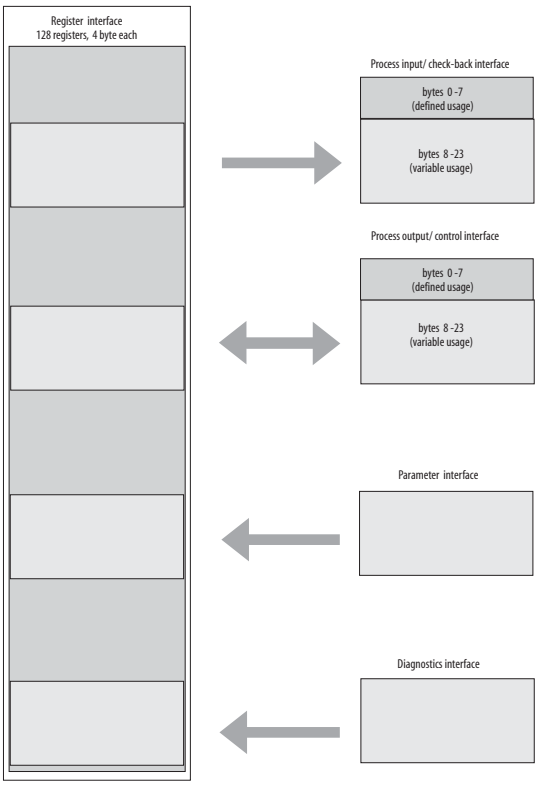



*Figure 3-1: Schematic representation of the register mapping*

### 3.1.2 Structure of the process data

The process data of the BL20-E-2CNT-2PWM contain

- 24 byte process input data, → see page 3-14
- 24 byte process output data, → see page 3-19

Additionally, the module provides

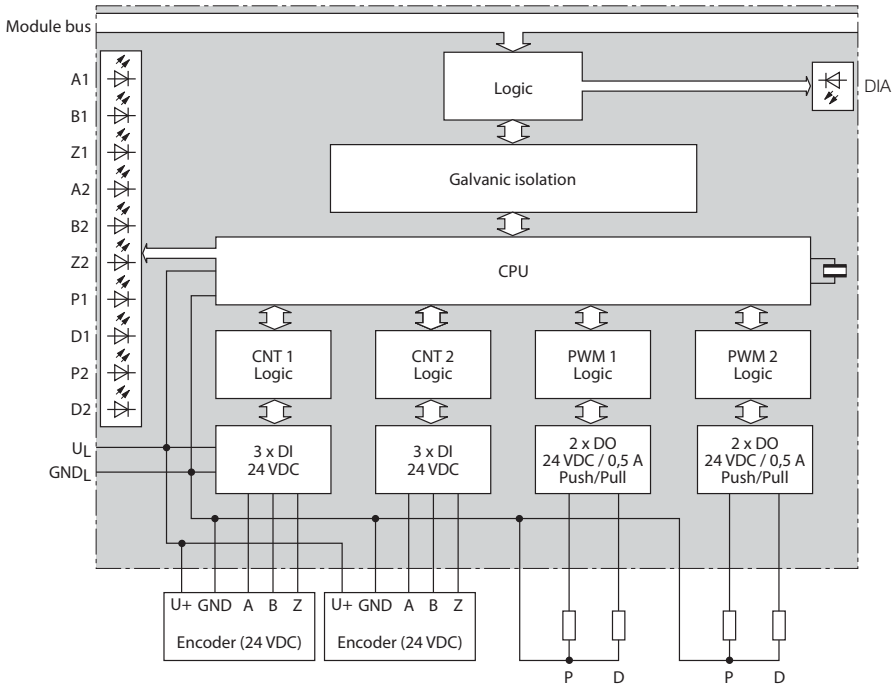
- 4 byte diagnostic data, → see page 3-9
  and
- 16 byte parameter data, → see page 3-10

## 3.2 Technical properties

*Figure 3-2: BL20-E-2CNT-2PWM*

### 3.2.1 Block diagram

*Figure 3-3: block diagram*



### 3.2.2 Technical data

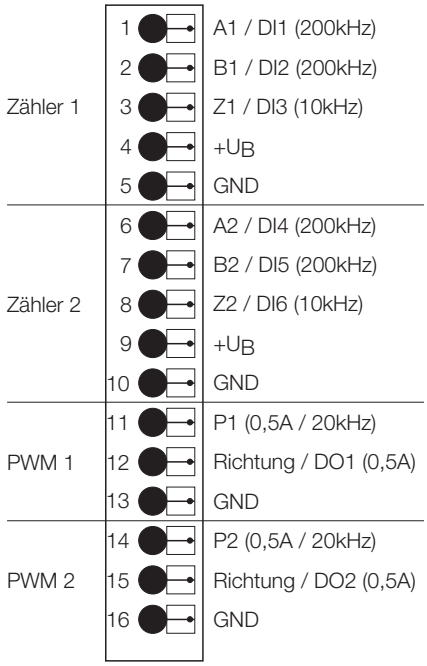
*Table 3-3: technical data*

| Designation | BL20-E-2CNT-2PWM | |
|---|---|---|
| Number of channels | | |
| count inputs | 2 | |
| PWM outputs | 2 | |
| Nominal voltage from supply terminal ($U_L$) | 24 VDC | |
| Nominal current from supply terminal ($I_L$) | typ. 35 mA all in- and outputs are ‚zero' | |
| Nominal current consumption from module bus ($I_{MB}$) | < 30 mA | |
| Power loss of the module ($P_V$) | < 2 W | |
| Counter function | | |
| Sensor supply | | |
| Output voltage | UL ( 24VDC ) | |
| Output current | < 0.5 A, not protected | |
| **Digital inputs for count signals A, B Z** | | |
| Input voltage | 0 to 30 VDC | |
| Parameterizable switching threshold $U_{SE}$ | **2.5 V** | **7.5 V** |
| Low level $I_L$ (active) | 0 to 1 V | 0 to 4.5 V |
| High level $I_{HL}$ (active) | 3.5 to 30 V | 7.5 to 30 V |
| Input current | | |
| Low level $I_L$ (active) | 0 to 0.1 mA | 0 to 0.4 mA |
| High level $I_{HL}$ (active) | 0.3 to 3 mA | 0.6 to 3 mA |
| Frequency (f) | | |
| A | max. 200 kHz | |
| B | max. 200 kHz | |
| Z | max. 10 kHz | |
| Minimum pulse width (maximum counting frequency) | | |
| at 200 kHz | ≥ 2,5 µs | |
| at 31.25 kHz | ≥ 16 µs | |

| Pulse- and direction output Px, Dx | |
|---|---|
| $R_{ON}$ switch-on resistance | 300 mΩ |
| Output current $I_A$ | |
| High level (nominal value) | 0.5 mA |
| High level $I_{AMAX}$ | 0.6 A (according to IEC 6 1131-2) |
| Simultaneity factor | 100 % |
| Switching frequency | |
| at ohmic load | 20 kHz |
| short-circuit proof | Yes |
| Isolation voltage | |
| $U_{TMB}$ (module bus/ IOs) | 500 $V_{eff}$ |
| $U_{FE}$ (module bus or field/ FE) | 500 $V_{eff}$ |
| Measurement ranges | |
| Count mode (all modes) | up to 200 kHz |
| frequency measurement | up to 200 kHz |
| period duration measurement | up to 178 s |

### 3.2.3 wiring diagram

*Figure 3-4: Pin assignment BL20-E-2CNT-2PWM*



### 3.2.4 Diagnostic/ and status messages

*Table 3-4: LED-displays*

| LED | Display | Meaning | Remedy |
|---|---|---|---|
| DIA | red, flashing, 0.5 Hz | Diagnosis pending | |
| | red, | Module bus communication failure | Check if more than two adjoining electronics modules have been pulled.<br>Check the power supply to the module bus. |
| | OFF | No error message or diagnosis | – |
| Ax, Bx, Zx | green | Input active | - |
| | OFF | Inputs not active | |
| Px, Dx | green | Output active | - |
| | red, | Overload at output | - |
| | OFF | Output inactive | |

### 3.2.5 Diagnostic data of the module

The module's diagnostic data contain error messages that are operation and application relevant for the control system. 4 bytes are used to transfer the diagnostic data.

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | HW_ERR | CNT1_ PAR_ERR | X | X | X | X | X | X |
| 1 | HW_ERR | CNT2_ PAR_ERR | X | X | X | X | X | X |
| 2 | HW_ERR | PWM1_ PAR_ERR | X | X | X | X | P1_DIAG | D1_DIAG |
| 3 | HW_ERR | PWM2_ PAR_ERR | X | X | X | X | P2_DIAG | D2_DIAG |

*Table 3-5: Diagnostics of the BL20-E-2CNT-2PWM*

| Diagnostic message | Values | Meaning |
|---|---|---|
| CNT1_PAR_ERR, CNT2_PAR_ERR, PWM1_PAR_ERR, PWM2_PAR_ERR | 0 | Parameter set of function unit correct |
| | 1 | Faulty / inconsistent parameters, wrong parameterization |
| P1_DIAG, P2_DIAG, D1_DIAG, D2_DIAG | 0 | No diagnostic message |
| | 1 | Diagnosis pending at channel (short circuit) |
| HW_ERR | 0 | No diagnostic message |
| | 1 | "Hardware error"<br>Display of common errors of the module's hardware (e.g. CRC-error, adjustment error….).<br>Change of device necessary. |

### 3.2.6 Parameter data of the module

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | input A1 | input B1 | input Z1 | X | diag-nostic CNT1 | measure-ment mode CNT1 | main count direction CNT1 | |
| 1 | filter Z1 | | filter A1, B1 | | X | pull up Z1 | X | threshold input A,B,Z CNT1 |
| 2 | mode Z1 | | | | mode CNT1 | | | |
| 3 | input A2 | input B2 | input Z2 | X | diag-nostic | measure-ment mode CNT2 | main count direction CNT2 | |
| 4 | filter Z2 | | filter A2, B2 | | X | pull up Z2 | X | threshold input A,B,Z CNT2 |
| 5 | mode Z2 | | | | mode CNT2 | | | |
| 6 | diag-nostic PWM1 | X | mode D1 | | | | | |
| 7 | DBP1 STS MODE | | substi-tute value P1 | substi-tute value D1 | mode PWM1 | | | |
| 8 | diag-nostic PWM2 | X | mode D2 | | | | | |
| 9 | DBP2 STS MODE | | substi-tute value P2 | substi-tute value D2 | mode PWM2 | | | |
| 10 | X | ADR AUX REG1 RD DATA | | | | | | |
| 11 | X | ADR AUX REG2 RD DATA | | | | | | |
| 12 | X | ADR AUX REG3 RD DATA | | | | | | |
| 13 | X | ADR AUX REG1 WR DATA | | | | | | |
| 14 | X | ADR AUX REG2 WR DATA | | | | | | |
| 15 | X | ADR AUX REG3 WR DATA | | | | | | |

X = reserved

*Table 3-6: Parameters of the BL20-E-2CNT-2PWM*

**A** *Default-setting*

| Parameter name | Value | Meaning | |
|---|---|---|---|
| Main count direction CNTx | 00 | Basic function **A** | |
| | 01 | None | |
| | 10 | Up | |
| | 11 | Down | |
| Measurement mode CNTx | 0 | Frequency measurement **A** | |
| | 1 | period duration measurement | |
| Diagnostic CNTx, Diagnostic PWMx | 0 | Diagnostic messages of the function unit activated in diagnostic interface **A** | |
| | 1 | Diagnostic messages of the function unit deactivated in diagnostic interface A | |
| Input Ax, Input Bx, Input Zx, | 0 **A** | Signal logic remains (LOW = 0 / HIGH = 1) | |
| | 1 | Invert signal before processing | |
| Threshold input A,B,Z CNTx | 0 **A** | Threshold 7.5V (only valid for Ax, Bx, Zx) | |
| | 1 | Threshold 2.5V (only valid for Ax, Bx, Zx) | |
| Pull Up Zx | 0 **A** | Pull Up resistance 20 kΩ off | |
| | 1 | Pull Up resistance 20 kΩ on | |
| Filter Ax, Bx | 00 | 2 µs **A** | Irrespective of the setting for the filter property, the maximum input frequency of the channel has to be considered |
| | 01 | 16 µs | |
| | 10 | reserved | |
| | 11 | | |
| Filter Zx | 00 | 2 µs **A** | Irrespective of the setting for the filter property, the maximum input frequency of the channel has to be considered |
| | 01 | 16 µs | |
| | 10 | reserved | |
| | 11 | | |

*Table 3-6: Parameters of the BL20-E-2CNT-2PWM*

| Parameter name | Value | Meaning |
|---|---|---|
| Mode CNTx (→ see page 4-9) | 0000 **A** | Pulse direction, single sample |
| | 0001 | Pulse direction, double sample |
| | 0010 | AB mode, single sample |
| | 0011 | AB mode, double sample |
| | 0100 | AB mode, four samples |
| | 0101 to 1110 | reserved |
| | 1111 | AB only input |
| Mode Zx (CNT1 page 4-12, PWM1 page 5-11) | 0000 | Alarm input CNT |
| | 0001 **A** | HW gate CNT |
| | 0010 | Single Latch-Retrigger CNT |
| | 0011 | Continuous latch retrigger CNT |
| | 0100 | Single L.-R. and HW gate CNT |
| | 0101 | Continuous L.-R. and HW gate CNT |
| | 0110 | reserved |
| | 0111 | Alarm input PWM |
| | 1000 | HW gate PWM |
| | 1001 | Retrigger PWM |
| | 1010 to 1110 | reserved |
| | 1111 | Z just input |
| mode Dx (→ see page 6-2) | | Definition of the function for Dx (default = 11 1111 → single output, can be controlled via process data) |
| Mode PWMx (→ see page 5-2) | 0000 **A** | PD DC Definition: |
| | 0001 | HT LT Definition |
| | 0010 to 0111 | reserved |
| | 1111 | P just output |
| Substitute value Px, Dx | 0 **A** | The output of the substitute value depends on the parameterization of the used gateway (→ documentation for the BL20-gateways). |
| | 1 | |

*Table 3-6: Parameters of the BL20-E-2CNT-2PWM*

| Parameter name | Value | Meaning |
|---|---|---|
| DBPx STS MODE | 00 **A** | STS_DBPx = 1<br>with<br>(REG_CNTx_CMP0) <= (REG_CNTx_CNT) < (REG_CNTx_CMP1) |
| | 01 | reserved |
| | 10 | |
| | 11 | STS_DBPx = Px |
| ADR AUX REGx WR DATA | | Address of the basic write registers<br>(Default:<br>ADR AUX REG1 WR DATA = 0x60, ADR AUX REG2 WR DATA = 0x61, ADR AUX REG3 WR DATA = 0x70) |
| ADR AUX REGx RD DATA | | Address of the basic read registers<br>(Default:<br>ADR AUX REG1 RD DATA = 0x20, ADR AUX REG2 RD DATA = 0x21, ADR AUX REG3 RD DATA = 0x40) |

### 3.2.7 Process data of the module

**Process input/ check-back interface**

| | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| CNTx (page 3-15) | 0 | A1 | B1 | Z1 | STS_CNT1_DIR | STS_CNT1_LOGMSG | STS_CNT1_SFKT_EN | STS_CNT1_RUN | STS_CNT1_GENERAL_EN |
| | 1 | MSG_CNT1_SW_LR | MSG_CNT1_SFKT | MSG_CNT1_FQE | MSG_CNT1_ND | MSG_CNT1_OFLW | MSG_CNT1_UFLW | MSG_CNT1_CMP1 | MSG_CNT1_CMP0 |
| | 2 | A2 | B2 | Z2 | STS_CNT2_DIR | STS_CNT2_LOGMSG | STS_CNT2_SFKT_EN | STS_CNT2_RUN | STS_CNT2_GENERAL_EN |
| | 3 | MSG_CNT2_SW_LR | MSG_CNT2_SFKT | MSG_CNT2_FQE | MSG_CNT2_ND | MSG_CNT2_OFLW | MSG_CNT2_UFLW | MSG_CNT2_CMP1 | MSG_CNT2_CMP0 |
| PWMx page 3-16 | 4 | STS_PWM1_LOGMSG | STS_PWM1_SFKT_EN | STS_PWM1_RUN | STS_PWM1_GENERAL_EN | MSG_PWM1_DO_ERR | MSG_PWM1_SFKT | MSG_PWM1_NDDC | MSG_PWM1_SW_LR |
| | 5 | STS_PWM2_LOGMSG | STS_PWM2_SFKT_EN | STS_PWM2_RUN | STS_PWM2_GENERAL_EN | MSG_PWM2_DO_ERR | MSG_PWM2_SFKT | MSG_PWM2_NDDC | MSG_PWM2_SW_LR |
| Communication page 3-17 | 6 | REG_WR_ACCEPT | REG_WR_AKN | REG_RD_ABORT | STS_CONFIG_ERR | STS_DBP2 | D2 | STS_DBP1 | D1 |
| | 7 | X | REG_RD_ADR | | | | | | |
| User data page 3-18 | 8 | REG_RD_DATA, Byte 0 | | | | | | | |
| | 9 | REG_RD_DATA, Byte 1 | | | | | | | |
| | 10 | REG_RD_DATA, Byte 2 | | | | | | | |
| | 11 | REG_RD_DATA, Byte 3 | | | | | | | |
| | 12 | AUX_REG1_RD_DATA, Byte 0 | | | | | | | |
| | ... | ... | | | | | | | |
| | 15 | AUX_REG1_RD_DATA, Byte 3 | | | | | | | |
| | 16 | AUX_REG2_RD_DATA, Byte 0 | | | | | | | |
| | ... | ... | | | | | | | |
| | 19 | AUX_REG2_RD_DATA, Byte 3 | | | | | | | |
| | 20 | AUX_REG3_RD_DATA, Byte 0 | | | | | | | |
| | ... | ... | | | | | | | |
| | 23 | AUX_REG3_RD_DATA, Byte 3 | | | | | | | |

X = reserved

**Note**

STATUS- (STS) or error messages (ERR) are volatile messages which are reset due to a change in status or due to the elimination of an error. In contrast, MSG describes a**non volatile** flag, which is set due to a certain event. It has to be reset (→ see Resetting the control bits (page 8-4)).

*Table 3-7: check-back interface*

| Bit | Value | Meaning |
|---|---|---|
| **CNTx** | | |
| STS_CNTx_GENERAL_EN | 0 | Function (CNTx) disabled |
| | 1 | Function enabled |
| STS_CNTx_RUN | 0 | CNTx: Counter not ready to count |
| | 1 | CNTx: Counter ready to count |
| STS_CNTx_SFKT_EN | 0 | Special function of Z disabled for CNTx |
| | 1 | Special function of Z enabled for CNTx |
| STS_CNTx_LOGMSG | 0 | Current status of MSG bits |
| | 1 | Status of MSG bits are frozen |
| STS_CNTx_DIR | 0 | CNTx: Counter counts down. |
| | 1 | CNTx: Counter counts up. |
| Ax, Bx, Zx | 0 | Digital input is LOW. |
| | 1 | Digital input is HIGH. |
| MSG_CNTx_CMP0 | 0 | No message active that reports that the compare value CMP0 has been reached. |
| | 1 | The counter CNTx reports that the compare value CMP0 was reached. |
| MSG_CNTx_CMP1 | 0 | No message active that reports that the compare value CMP1 has been reached. |
| | 1 | The counter CNTx reports that the compare value CMP1 was reached. |
| MSG_CNTx_UFLW | 0 | No message active that reports that the lower count limit has been reached. |
| | 1 | The counter CNTx reports the lower count limit was reached. |

*Table 3-7: check-back interface*

| Bit | Value | Meaning |
|---|---|---|
| MSG_CNTx_OFLW | 0 | No message active that reports that the upper count limit has been reached. |
| | 1 | The counter CNTx reports the upper count limit was reached. |
| MSG_CNTx_ND | 0 | No message active that reports a zero crossing. |
| | 1 | The counter CNTx reports a zero crossing. |
| MSG_CNTx_FQE | 0 | No error occurred in frequency or period duration measurement. |
| | 1 | The counter CNTx reports an error in frequency/ period duration measurement.<br>Possible error causes:<br>Max. length of the no-pulse period reached.<br>The value cannot be displayed correctly in the register for the "pulses per integration time" REG_CNTx_IPI (page 9-7 or page 9-8) due to a multiplicator which has been set too high in register REG_CNTx_MUL (page 9-7 or page 9-8). |
| MSG_CNTx_SFKT | 0 | The event according to there parameterized special function CNT1_SFKT_DISABLE did **not** occur. |
| | 1 | The event according to there parameterized special function CNT1_SFKT_DISABLEoccurred. |
| MSG_CNTx_SW_LR | 0 | The function Latch-Retrigger (→ see also page 4-16) has not been activated. |
| | 1 | The function Latch-Retrigger (→ see also page 4-16) has been activated via bit CNTx_SW_LR = 1 (→ see also page 3-20). |
| **PWMx** | | |
| MSG_PWM1x_SW_LR | 0 | The function Latch-Retrigger (→ see also page 4-16) has not been activated. |
| | 1 | The function Latch-Retrigger (→ see also page 4-16) has been activated via bit PWMx_SW_LR = 1 (→ see also page 3-21). |
| MSG_PWMx_NDDC | 0 | No message active that reports a zero crossing of the PWMx. |
| | 1 | The counter PWMx reports a zero crossing. |
| MSG_PWMx_SFKT | 0 | The event according to there parameterized special function PWMx_SFKT_DISABLE did **not** occur . |
| | 1 | The event according to there parameterized special function PWMx_SFKT_DISABLE occurred. |

*Table 3-7: check-back interface*

| Bit | Value | Meaning |
|---|---|---|
| MSG_PWMx_DO_ERR | 0 | No error message from outputs Px / Dx. |
| | 1 | One of the outputs Px (Px_DIAG) or Dx (Dx_DIAG) of the corresponding PWMx-channel sent an error. |
| STS_PWMx_GENERAL_EN | 0 | Function (PWMx) disabled |
| | 1 | Function enabled, with a change from 0 → 1 the channel is set to the initial state |
| STS_PWMx_RUN | 0 | PWMx-signal output not active |
| | 1 | PWMx-signal output active |
| STS_PWMx_SFKT_EN | 0 | Special function of Z disabled for PWMx |
| | 1 | Special function of Z enabled for PWMx |
| STS_PWMx_LOGMSG | 0 | Current status of MSG bits |
| | 1 | Status of MSG bits are frozen |
| **Communication** | | |
| Dx | 0 | Digital input is LOW |
| | 1 | Digital input is HIGH |
| STS_DBPx | 0 | Status of the information defined through DBPx STS MODE. |
| | 1 | |
| STS_CONFIG _ERR | 0 | The present configuration is OK. |
| | 1 | In REG_CONFIG_ERR an error is reported |
| REG_RD_ABORT | 0 | The reading of the register defined in REG_RD_ADR has been accepted and executed. The content of the register can be found in the user data (REG_RD_DATA). |
| | 1 | Reading of the register defined in REG_RD_ADR has not been accepted. The register content (REG_RD_DATA) is zero. |
| REG_WR_AKN | 0 | A change of register contents had been assigned through a process output. |
| | 1 | No change of register contents through a process output. (Write access REG_WR to the register interface is only possible, if this bit was zero before; handshake for data transfer to the registers). |

*Table 3-7: check-back interface*

| Bit | Value | Meaning |
|---|---|---|
| REG_WR_ACCEPT | 0 | Writing the user data from the control interface to the register addressed with REG_WR_ADR in the control interface could not be done. |
| | 1 | Writing the user data from the control interface to the register addressed with REG_WR_ADR in the control interface was successful. |
| REG_RD_ADR | 0 to 127 | Address of the input register of which the content is shown in the user data (REG_RD_DATA) in the check-back interface if REG_RD_ABORT = 0. |
| **User data** | | |
| REG_RD_DATA | 0 ... $2^{32}$-1 | Content of the register of which the address is transferred in the process input data (REG_RD_ADR) if REG_RD_ABORT = 0. If not, REG_RD_DATA = 0. |
| AUX_REGx_RD_DATA | 0 ... $2^{32}$-1 | Value, which is read from the register with the address defined in the parameterization in ADR_AUX_REGx_RD_DATA. |

### Process output/ control interface

| | | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Control bytes | CNT x (page 3-20) | 0 | X | CNT1_ SINGLE | CNT1_ SW_LR | CNT1_SFKT _DISABLE | X | CNT1_ LOGMSG | CNT1_ ENABLE | CNT1_ GENERAL_ DISABLE |
| | | 1 | X | CNT2_ SINGLE | CNT2_ SW_LR | CNT2_SFKT _DISABLE | X | CNT2_ LOGMSG | CNT2_ ENABLE | CNT2_ GENERAL_ DISABLE |
| | PWM x (page 3-20) | 2 | X | PWM1_ SINGLE | PWM1_ SW_LR | PWM1_ SFKT_ DISABLE | X | PWM1_ LOGMSG | PWM1_ ENABLE | PWM1_ GENERAL_ DISABLE |
| | | 3 | X | PWM2_ SINGLE | PWM2_ SW_LR | PWM2_ SFKT_ DISABLE | X | PWM2_ LOGMSG | PWM2_ ENABLE | PWM2_ GENERAL_ DISABLE |
| | DOs | 4 | X | X | SET_P2 | SET_ D2 | X | X | SET_P1 | SET_ D1 |
| register access (page 9-2) | | 5 | REG_WR | X | X | X | X | AUX_REG3_ WR_EN | AUX_REG2_ WR_EN | AUX_REG1_ WR_EN |
| | | 6 | X | REG_WR_ADR | | | | | | |
| | | 7 | X | REG_RD_ADR | | | | | | |
| User data | | 8 | REG_WR_DATA, byte 0 | | | | | | | |
| | | 9 | REG_WR_DATA, byte 1 | | | | | | | |
| | | 10 | REG_WR_DATA, byte 2 | | | | | | | |
| | | 11 | REG_WR_DATA, byte 3 | | | | | | | |
| | | 12 | AUX_REG1_WR_DATA, byte 0 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 15 | AUX_REG1_WR_DATA, byte 3 | | | | | | | |
| | | 16 | AUX_REG2_WR_DATA, byte 0 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 19 | AUX_REG2_WR_DATA, byte 3 | | | | | | | |
| | | 20 | AUX_REG3_WR_DATA, byte 0 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 23 | AUX_REG3_WR_DATA, byte 3 | | | | | | | |

X = reserved

*Table 3-8: Process output data of the module*

**A** *Default-setting*

| Bit | Value | Meaning |
|---|---|---|
| CNTx_GENERAL_ DISABLE | 0 | Enable function unit CNTx |
| | 1 | Disable function unit CNTx generally |
| CNTx_ENABLE | 0 | Not activated |
| | 1 | Enable counter CNTx (SW gate)<br>The enable is done either per SW- **or** per HW gate, → see Enabling the counter (page 4-4)). |
| CNTx_LOGMSG | 0 | The messages in the MSG-bits (MSG for CNTx (page 8-2)) in the Process input/ check-back interface are active. |
| | 1 | With a change from 0 → 1 the MSG data are held and actual incoming messages are stored to register REG_PWMx_LOGMSG, → see for example Error handling in the control interface/ check-back interface (page 8-1).<br>Before switching to REG_CNTx_LOGMSG, this register is set to "0".<br>With a change from 1 → 0, all data from REG_CNTx_LOGMSG are copied to the MSG-bits in the Process input/ check-back interface. |
| CNT1_SFKT_DISABLE | 0 | Enable the special function of input Zx depending on the parameterization Mode Zx. |
| | 1 | Disable the special function of input Zx. |
| CNTx_SW_LR | 0 | Not activated |
| | 1 | A Software (SW) latch retrigger (page 4-5) has to be executed at counter CNTx with a change from 0 → 1. |
| CNTx _SINGLE | 0 | Continuous enabling of CNTx (Method of counting: periodical counting (page 4-7)) |
| | 1 | Single enabling of CNTx (Method of counting: single counting (page 4-6)) |
| PWMx_GENERAL_ DISABLE | 0 | Enable function unit PWMx |
| | 1 | Disable function unit PWMx |
| PWMx_ ENABLE | 0 | Not activated |
| | 1 | Enable output PWMx<br>(The enable is done either per SW- **or** per HW gate, → see Enabling the counter (page 4-4)). |

*Table 3-8: Process output data of the module*

| Bit | Value | Meaning |
|---|---|---|
| PWMx_LOGMSG | 0 | The messages in the MSG-bits (MSG for PWMx (page 8-3)) in the Process input/ check-back interface are active. |
| | 1 | With a change from 0 → 1 the MSG data are held and actual incoming messages are stored to register REG_PWMx_LOGMSG, → see for example Error handling in the control interface/ check-back interface (page 8-1). Before switching to REG_PWMx_LOGMSG, this register is set to "0". With a change from →, all data from REG_CNTx_LOGMSG are copied to the MSG-bits in the Process input/ check-back interface. |
| PWMx_SFKT_DISABLE | 0 | Enable the special function of input Zx depending on the parameterization. |
| | 1 | Disable the special function of input Zx depending on the parameterization. |
| PWMx_SW_LR | 0 | Not activated |
| | 1 | A latch retrigger has to be executed at counter PWMx with a change from 0 → 1. |
| PWMx_SINGLE | 0 | Continuous enabling of PWM |
| | 1 | Single enabling of PWMx |
| SET_ Dx | 0 | Clear bit Dx |
| | 1 | Set bit Dx |
| SET_ Px | 0 | Clear bit Px |
| | 1 | Set bit Px |
| AUX_REG1_ WR_EN<br>...<br>AUX_REG3_ WR_EN | 0 | Disabling the writing of register data with the register contents in AUX_REGx_WR_DATA. This option avoids an unintentional writing to registers in the Register interface (→ see also Internal registers - reading and writing (page 9-2)). |
| | 1 | Writing of the Register interface with the register contents in AUX_REGx_WR_DATA is enabled (→ see also Internal registers - reading and writing (page 9-2)). |
| REG_WR | 0 | Initial state |
| | 1 | Triggering a write command. The register of which the address has been defined with REG_WR_ADR, will be written with data from REG_WR_DATA. |
| REG_WR_ADR | 0...127 | Address of the register, which has to be written with REG_WR_DATA (→ see below). |

*Table 3-8: Process output data of the module*

| Bit | Value | Meaning |
|---|---|---|
| REG_RD_ADR | 0...127 | Address of the register to be read. With a RD_ABORT = 0 in REG_RD_DATA, the user data can be found in the Process input/ check-back interface (page 3-14). |
| REG_WR_DATA, Byte 0<br>...<br>REG_WR_DATA, Byte 3 | 0 $2^{32}$-1 | Value which, during a write operation, has to be written to the register selected with REG_WR_ADR (→ see above). |
| AUX_REGx_WR_DATA, Byte 0<br>...<br>AUX_REG1_WR_DATA, byte 3 | 0 $2^{32}$-1 | Value which, during a write operation, has to be written to the register defined in (ADR AUX REGx WR DATA (page 3-13)) in the parameterization. |

# 4 Functions of the count inputs (CNT1 and CNT2)

## 4.1 Basic functions of the count inputs CNTx

### 4.1.1 Load count value

The count value can be loaded directly via the Register interface (page 9-4). To do so, the register (REG_CNTx_CNT) has to be written directly with the desired count value using the access via the Process output/ control interface (page 3-19).

| Register name | Register no. | Max. value | Min. value | Default value |
|---|---|---|---|---|
| REG_CNT1_CNT | 32<br>(0×20) | +2147483647<br>(0×7F FF FF FF) | -2147483648<br>(0×80 00 00 00) | 0<br>(0×00 00 00 00) |
| REG_CNT2_CNT | 64<br>(0×40) | | | |

**Note**

If the value to be loaded is outside the count limits, the value is not transferred as count value, but the count limit which is the closest to the value is loaded instead. No error message is set in REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

### 4.1.2 Load load value

The load value is loaded via the Process output/ control interface (page 3-19) and copied into the count value (REG_CNTx_CNT) due to certain events.

These events have to be defined via the Parameter data of the module (page 3-10).

| Register name | Register no. | Max. value | Min. value | Default value |
|---|---|---|---|---|
| REG_CNT1_LOAD VAL | 35<br>(0×23) | +2147483647<br>(0×7F FF FF FF) | -2147483648<br>(0×80 00 00 00) | 0<br>(0×00 00 00 00) |
| REG_CNT2_LOAD VAL | 67<br>(0×43) | | | |

**Note**

If a load value is loaded which is outside the count limits, the value is transferred and an error message is generated in REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

### 4.1.3 Set count limits

The count limits are contained in the following registers:

| Register name | Register no. | Max. value | Min. value | Default value |
|---|---|---|---|---|
| REG_CNT1_LOLIMIT<br>lower count limit CNT1 | 36<br>(0×24) | +2147483647<br>(0×7F FF FF FF) | -2147483648<br>(0×80 00 00 00) | -2147483648<br>(0×80 00 00 00) |
| REG_CNT1_HILIMIT<br>upper count limit CNT1 | 37<br>(0×25) | | | +2147483647<br>(0×7F FF FF FF) |
| REG_CNT2_LOLIMIT<br>lower count limit CNT2 | 68<br>(0×44) | | | +2147483647<br>(0×7F FF FF FF) |
| REG_CNT2_HILIMIT<br>upper count limit CNT2 | 69<br>(0×45) | | | -2147483648<br>(0×80 00 00 00) |

The count limits can be set via the Process output/ control interface (page 3-19).

**Note**

If the count limits are loaded with (REG_CNTx_HILIMIT) < = (REG_CNTx_LOLIMIT), the value is transferred and an error message is recorded in the REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

If a count limit is set so that the actual count value is outside the count range, the current value is set to the count limit which is closest to it. In this case *no* error message is recorded in register REG_CONFIG_ERRSTS.

### 4.1.4 Enabling the counter

**Prerequisite**:

Enabling the counter via hard- or software gate requires a general enable of the count function using CNT1_ GENERAL_ DISABLE = 0 (default-setting).

**Note**

The counter can be enabled either per software **or** per hardware gate:

Example:

**NOT** CNTx_GENERAL_DISABLE (page 3-19)

**AND** (SW-Tor **OR** HW-Tor)

**Hardware gate (HW gate)**

The count operation is enabled with signal Zx = 1 and disabled with Zx = 0.

**Parameterization of Zx**

In order to use Zx to enable the count operation, the special function of the input Zx has to be enabled as well
(→ see Special function Zx (CNT): HW gate (page 4-14))

The following applies for the HW gate:

STS_CNTx_RUN = 1,

if,

CNTx_GENERAL_DISABLE = 0
and

CNTx_SFKT_DISABLE = 0 and

Zx = 1

**Software-gate (SW gate)**

The counter enable is done by setting the bits CNT1_ ENABLE (CNT1) or CNT2_ ENABLE (CNT2) of the Process output/ control interface (page 3-19).

The following applies for the HW gate:

STS_CNTx_RUN = 1,

if,

CNTx_GENERAL_DISABLE = 0 and

CNTx_ENABLE = 1

### 4.1.5 Latch Retrigger (CNT)

The internal count value is re triggered, which means the current count value is stored, the load value is reloaded as count value and the count operation continues.

The count function has to be enabled (→ see Enabling the counter (page 4-4)) in order to execute this latch retrigger.

**Hardware (HW) latch retrigger**

The hardware latch retrigger is done via a signal change at Zx 0 → 1 (→ siehe Special function Zx (CNT): Synchronization (HW latch retrigger) (page 4-16)). The event is reported via MSGx_CNTx_SFKT of the Process input/ check-back interface (page 3-14) (bit 6 in byte 1 (CNT1) and byte 3 (CNT2)).

**Software (SW) latch retrigger**

The software latch retrigger is done via a signal change 0 → 1 in bit CNTx_SW_LR in the Process output/ control interface (page 3-19). The executed SW latch retrigger is confirmed through bit MSG_CNTx_SW_LR of the Process input/ check-back interface (page 3-14) (bit 7 in byte 1 (CNT1) and byte 3 (CNT2)).

### 4.1.6 Function of the CMPx compare registers

Each counter provides two compare registers.

If a counter reaches a count value which matches the content of one of its CMP-registers, this event is reported in the Process input/ check-back interface (page 3-14) through MSG_CNTx_CMP0 or respectively MSG_CNTx_CMP1.

This message remains active until it is reset via the control interface when reading the counter's status messages setting CNTx_LOGMSG 1 → 0.

The compare values are loaded via the control interface.

| Register name | Register no. | Max. value | Min. value | Default value |
|---|---|---|---|---|
| REG_CNT1_CMP0<br>Compare value 0 CNT1 | 38<br>(0×26) | +2147483647<br>(0×7F FF FF FF) | -2147483648<br>(0×80 00 00 00) | -2147483648<br>(0×80 00 00 00) |
| REG_CNT1_CMP1<br>Compare value 1 CNT1 | 39<br>(0×27) | | | +2147483647<br>(0×7F FF FF FF) |
| REG_CNT2_CMP0<br>Compare value 0 CNT2 | 70<br>(0×46) | | | +2147483647<br>(0×7F FF FF FF) |
| REG_CNT2_CMP1<br>Compare value 1 CNT2 | 71<br>(0×47) | | | -2147483648<br>(0×80 00 00 00) |

**Note**

If a compare value is loaded which is outside the count limits, the value is transferred and an error message is generated in REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

### 4.1.7 Method of counting: single counting

If, in the Process output/ control interface (page 3-19), single counting is activated via CNTx_SINGLE = **1**, the following events reset the status bit STS_CNTx_RUN page 3-14.

Depending on the main count direction definition (Parameter data of the module (page 3-10)), the counter reacts as follows:

*Table 4-1: Main count direction at CNTx_SINGLE = **1***

| **Main count direction CNTx** | | | | |
|---|---|---|---|---|
| **Bit 1** | **Bit 0** | | | |
| 0 | 0 | Basic function | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the lower count limit. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the upper count limit. |
| | | | Internal enable | The internal enable is automatically reset, if the counter value jumps to the other count limit after having exceeded the count limit. |
| 0 | 1 | None | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the load value. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the load value. |
| | | | Internal enable | The internal enable is automatically reset, if the counter value jumps to the load value after having exceeded the count limit. |
| 1 | 0 | Up | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the load value. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the upper count limit. |
| | | | Internal enabling | The internal enable is automatically reset, if the counter value jumps to the load value or the other count limit after having exceeded the count limit. |

*Table 4-1: Main count direction at CNTx_SINGLE = **1***

| Main count direction CNTx | | | | |
|---|---|---|---|---|
| **Bit 1** | **Bit 0** | | | |
| 1 | 1 | Down | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the lower count limit. The counter continues counting starting at the lower count limit. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the load value. The counter continues counting starting at the load value. |
| | | | Internal enable | The internal enable is not automatically reset. The counter remains active. |

The internal enable is reset automatically when a count limit is reached and STS_CNTx_RUN is set = 0. In this case, with CNTx_ENABLE = 1 being active, a counter restart is executed through first setting bit CNTx_GENERAL_DISABLE to "1" and then resetting it to "0".

### 4.1.8 Method of counting: periodical counting

If, in the Process output/ control interface (page 3-19), periodical counting is activated via CNTx_SINGLE = **0**, the counter reacts as follows depending on the main count direction defined (Parameter data of the module (page 3-10)).

*Table 4-2: Main count direction at CNTx_SINGLE = **0***

| Main count direction CNTx | | | | |
|---|---|---|---|---|
| **Bit 1** | **Bit 0** | | | |
| 0 | 0 | Basic function | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the lower count limit. The counter continues counting starting at the lower count limit. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the load value. The counter continues counting starting at the load value. |
| | | | Internal enabling | The internal enable is not automatically reset. The counter remains active. |

*Table 4-2: Main count direction at CNTx_SINGLE = **0***

| **Main count direction CNTx** | | | | |
|---|---|---|---|---|
| **Bit 1** | **Bit 0** | | | |
| 0 | 1 | None | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, when counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the load value and continues counting. |
| | | | Lower count limit | If, when counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the load value and continues counting. |
| | | | Internal enable | The internal enable is not automatically reset. The counter remains active. |
| 1 | 0 | Up | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the load value. The counter continues counting starting at the load value. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the upper count limit. The counter continues counting starting at the upper count limit. |
| | | | Internal enabling | The internal enable is not automatically reset. The counter remains active. |
| 1 | 1 | Down | Load value | The counting operation starts with the current count value. |
| | | | Upper count limit | If, while counting upwards, the counter reaches the upper count limit and another counting pulse is received, the counter value jumps to the lower count limit. The counter continues counting starting at the lower count limit. |
| | | | Lower count limit | If, while counting downwards, the counter reaches the lower count limit and another counting pulse is received, the counter value jumps to the load value. The counter continues counting starting at the load value. |
| | | | Internal enabling | The internal enable is not automatically reset. The counter remains active. |

**Note**

The counting operation starts with the current count value. The user can preset this value to a defined value (→ see Load load value (page 4-2)).

### 4.1.9 Count inputs Ax and Bx

The inputs A1, B1 are the inputs for counter 1 (CNT1), inputs A2, B2 are the inputs for counter 2 (CNT2).

The functions of these inputs and the counter operation mode must be set via the parameters

"mode CNT1"

and

""mode CNT2"

in the Parameter data of the module (page 3-10).

Possible functions (→ see page 4-9):

- Pulse and direction
  - Ax pulse, Bx direction/ single sample
  - Ax pulse, Bx direction/ double sample
- AB-operation mode:
  - Single sample
  - Double sample
  - Four samples
- Simple digital input

**Note**

The parameterization of undefined functions is reported through a diagnostic message with CNTx_PAR_ERR = 1 (→ see Diagnostic data of the module (page 3-9)).

IF CNTx_PAR_ERR = 1, THEN the counter can not be activated.

**Note**

If a parameter error occurs, this is reported in the diagnostic data and an error message is set in REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

### Operation mode: pulse and direction

If the function "pulse direction, single sample" (→ see Parameter data of the module (page 3-10)) is set, the rising or falling edges of input Ax are evaluated, depending on the parameterization of input Ax.

If the function "pulse direction, double sample" (→ see Parameter data of the module (page 3-10)) is set, the rising and falling edges of input Ax are evaluated, depending on the parameterization of input Ax.

Signal Bx defines the count direction.

*Figure 4-1: pulse and direction*

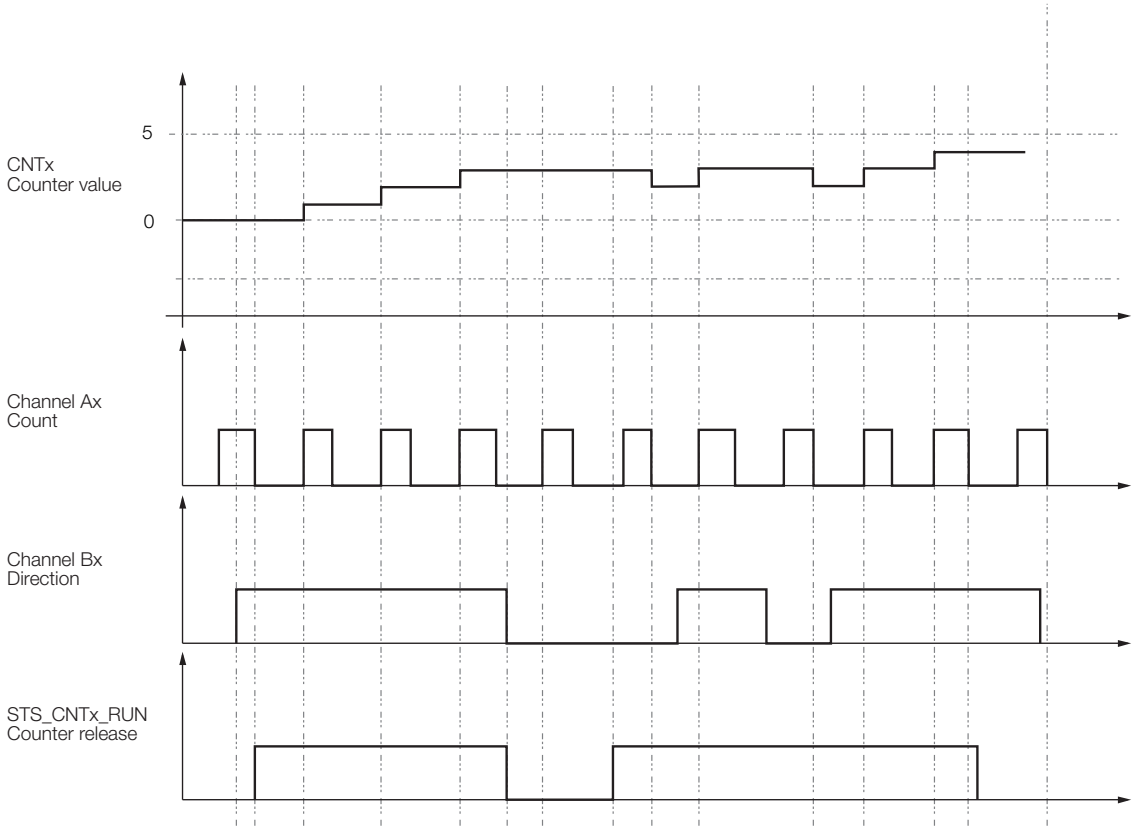
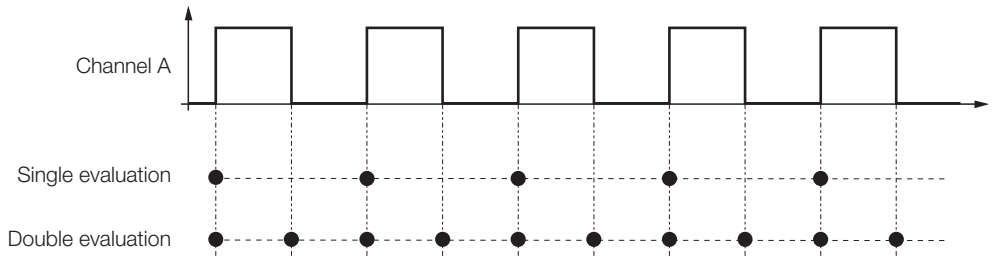


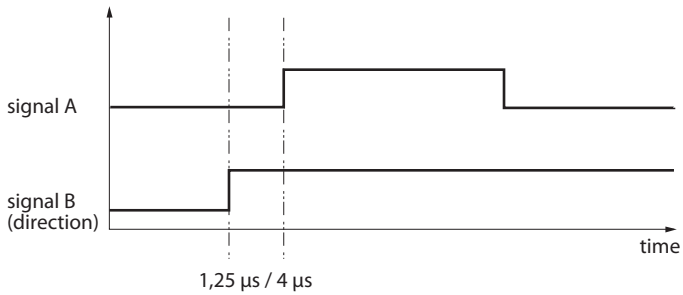

*Figure 4-2: pulse and direction, evaluation*



### Time span between direction signal (B) and count signal (A)

For pulse generators with a direction signal, it must be ensured that there is a gap of at least 1,25 µs or 4 µs between the direction signal (B) and the counter signal (A), depending on the input filter configured.

*Figure 4-3: Time span between direction signal (B) and count signal (A)*



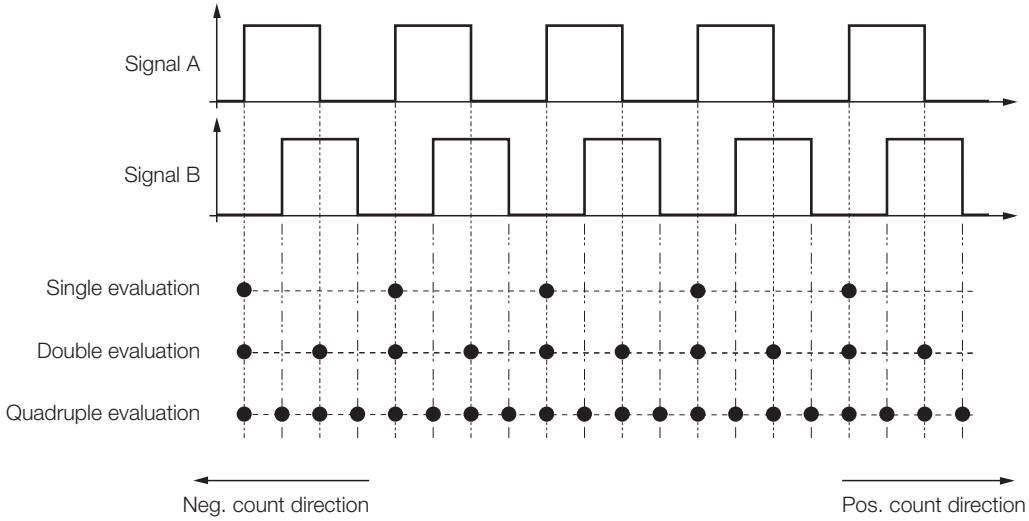

### Operation mode: AB-operation mode

In the AB mode, pulse and direction are determined by the phasing of the input signals Ax and Bx.

The counter counts upwards, if the signal sequence is run through from the left to the right (pos. count direction). The counter counts upwards, if the signal sequence is run through from the left to the right (neg. count direction).

The points in the following figure mark the scan points (change in count value) depending on the parameterization.

*Figure 4-4: AB-mode*



### 4.1.10 Special function of inputs Z1 and Z2 at CNTx

The inputs Z1 and Z2 can be used to support counter functions (or PWM functions, → see chapter 5).

**Note**

Each special function of Zx is **enabled** per default via CNTx_SFKT_DISABLE **= 0** in the Process output/ control interface (page 3-19). The enabling of this function is acknowledged via STS_CNTx_SFKT_EN = 1.
The special function is **disabled** via CNTx_SFKT_DISABLE **= 1**.

The function of Zx is defined through the parameter "mode Zx" (→ see Parameter data of the module (page 3-10))..

Possible functions of the count inputs (CNT1 and CNT2) at a signal change 0→ 1:

- alarm (→ see Special function Zx (CNT): alarm (page 4-13))
- HW gate (counter enabling, → see Special function Zx (CNT): HW gate (page 4-14))
- Synchronization (latch retrigger, single or periodical with the load value, → see Special function Zx (CNT): Synchronization (HW latch retrigger) (page 4-16))

Single synchronization *und* HW gate) is possible, using a **combination**of Z1 and Z2 (→ see also here Parameter data of the module (page 3-10)).

*Table 4-3: functions Z1 and Z2*

**A** *Default-setting*

| **Bit 7 ... Bit 4 (Value)** | **mode Z1** (→ see byte 2 of the Parameter data of the module (page 3-10)) | **mode Z2** (→ see byte 5 of the Parameter data of the module (page 3-10)) |
|---|---|---|
| 0000 | **Alarm input CNT** | |
| | A signal change 0→ 1 at Zx causes the setting of the MSG_CNTx_SFKT flag. This serves to also report short-time events. | |
| 0001 **A** | **HW gate CNT**<br>The enabling can also be done per software gate (→ see also here Enabling the counter (page 4-4)) | |
| | – with Zx = 0, the counter is disabled<br>– with Zx = 1, the counter is enabled | |
| 0010 | **Single Latch-Retrigger CNT** | |
| | With the first signal change 0 → 1 at Z1, the current count value is copied into the register REG_CNT1_LATCH and the load value 1 is transferred to counter 1. The count operation is not interrupted. | With the first signal change 0 → 1 at Z2, the current count value is copied into the register REG_CNT2_LATCH and the load value 2 is transferred to counter 2. The count operation is not interrupted. |
| 0011 | **Continuous latch retrigger CNT** | |
| | With the first signal change 0 → 1 at Z1, the current count value is copied into the register REG_CNT1_LATCH and the load value 1 is transferred to counter 1. The count operation is not interrupted. | With the first signal change 0 → 1 at Z2, the current count value is copied into the register REG_CNT2_LATCH and the load value 2 is transferred to counter 1. The count operation is not interrupted. |

*Table 4-3: functions Z1 and Z2*

| Bit 7 ... Bit 4 (Value) | mode Z1 (→ see byte 2 of the Parameter data of the module (page 3-10)) | mode Z2 (→ see byte 5 of the Parameter data of the module (page 3-10)) |
|---|---|---|
| 0100 | **Single L.-R. and HW gate CNT** | |
| | Single synchronization for CNT1 (Z1) **and** HW gate (Z2) for CNT1. The enable for Z1 and Z2 is done via CNT1_SFKT_EA .<br>– with the first signal change 0 → 1 at Z1, the current count value is copied into the register REG_CNT1_LATCH and the load value 1 is transferred to counter 1. The count operation is not interrupted.<br>– with Z2 = 0, CNT1 is disabled<br>– with Z2 = 1, CNT1 is enabled | Single synchronization for CNT2 (Z2) **and** HW gate (Z1) for CNT2. The enable for Z1 and Z2 is done via CNT2_SFKT_EA .<br>– with the first signal change 0→ at Z2, the current count value is copied into the register REG_CNT2_LATCH and the load value of CNT2 is transferred to counter 2. The count operation is not interrupted.<br>– with Z1 = 0, CNT2 is disabled<br>– with Z1 = 1, CNT2 is enabled |
| 0101 | **Continuous L.-R. and HW gate CNT** | |
| | Single synchronization for CNT1 (Z1) **and** HW gate (Z2) for CNT1. The enable for Z1 and Z2 is done via CNT1_SFKT_EA :<br>– with the first signal change 0 → 1 at Z1, the current count value is copied into the register REG_CNT1_LATCH and the load value 1 is transferred to counter 1. The count operation is not interrupted.<br>– with Z2 = 0, CNT1 is disabled<br>– with Z2 = 1, CNT1 is enabled | Periodical synchronization for CNT2 (Z2) **and** HW gate (Z1) for CNT2. The enable for Z1 and Z2 is done via CNT2_SFKT_EA :<br>– with the first signal change 0 → 1 at Z2, the current count value is copied into the register REG_CNT2_LATCH and the load value of CNT2 is transferred to counter 2. The count operation is not interrupted.<br>– with Z1 = 0, CNT2 is disabled<br>– with Z1 = 1, CNT2 is enabled |
| 0110 | reserved | |
| 0111 to 1001 | Functions of Zx for PWM1 and PWM2,<br>((→ see Special function of inputs Z1 and Z2 for the PWM (page 5-11)) | |
| 1010 to 1110 | reserved | |
| 1111 | **Z just input**<br>Simple digital input, the status is reported via the check-back interface. | |

**Special function Zx (CNT): alarm**

If Zx is parameterized as alarm signal for the counter (Parameter data of the module (page 3-10)), then the signal status is reported as message.

→ Enable/disable the special function (→ see page 4-12)

After an alarm signal occurred, bit MSG_CNTx_SFKT page 3-14 in the Process input/ check-back interface is set.

**Note**

When using alarm-signals as Open Collector, a pull up resistance can be switched via the Parameter data of the module (page 3-10).

### Special function Zx (CNT): HW gate

→ Enable/disable the special function (→ see page 4-12)

If Zx is parameterized as HW gate for the counter (Parameter data of the module (page 3-10)), the counter gate is
enabled with Zx = 1
and
disabled with Zx = 0.

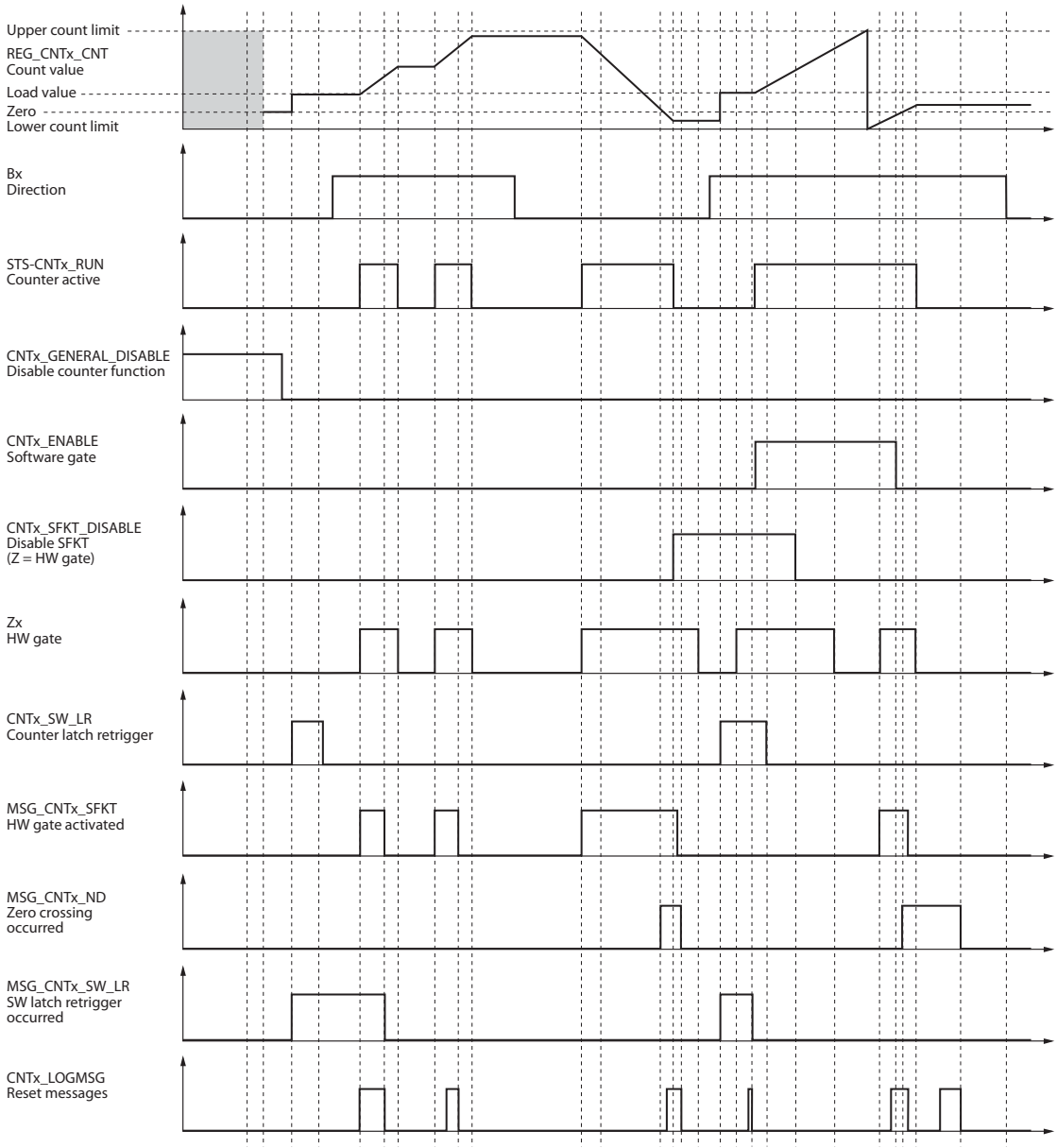
**Note**

Enabling the counter can be done either via the hardware **or** the software gate. Please read Enabling the counter (page 4-4).

After opening the HW gate, bit MSG_CNTx_SFKT page 3-14in the Process input/ check-back interface is set.

Opening and closing the counter gate via HW or SW only influences the counter enable. Count values are not influenced.

*Figure 4-5: Count function, Zx as HW gate*




Rule:
STS_CNTx_RUN = /CNTx_GENERAL_DISABLE & ((/CNTx_SKT_DISABLE)& Zx) I (CNTx_ENABLE))

### Special function Zx (CNT): Synchronization (HW latch retrigger)

→ Enable/disable the special function (→ see page 4-12)

If the synchronization of the counter value is parameterized for Zx (Parameter data of the module (page 3-10)), Zx is used as **hardware (HW) latch retrigger** .

With a signal change 0→ 1 at input Zx,

**1** the current count value is stored to REG_CNTx_LATCH page 9-8,

**2** the load value is transferred as count value from REG_CNTx_LOADVAL page 9-6 to REG_CNTx_CNTpage 9-7

and

**3** the count operation is continued.

After an latch retrigger occurred, the bit MSG_CNT1_SFKT (page 3-14) or resp. MSG_CNT2_SFKT (page 3-14) in the check-back interface is set. It can then be reset via bit CNTx_LOGMSG page 3-19 in the control interface with 0 → 1→ 0.

- Single synchronization:
  If in the Process output/ control interface (page 3-19) the single synchronization is parameterized with CNTx_SINGLE = 1 (CNT1: byte 0, bit 6/ CNT2: byte 1, bit 6), a latch retrigger is executed only with the **first** signal change 0 → 1 at Zx after enabling the counter with CNTx_SFKT_DISABLE = 0.
- Periodical synchronization:
  If in the Process output/ control interface (page 3-19) the single synchronization is parameterized with CNTx_SINGLE = 0 (CNT1: byte 0, bit 6/ CNT2: byte 1, bit 6), a latch retrigger is executed only with the **first** signal change 0 → 1 at Zx after enabling the counter with CNTx_SFKT_DISABLE = 0.

When executing a HW latch retrigger, the following applies:
(REG_CNTx_LATCH) = (REG_CNTx_CNT) and

(REG_CNTx_CNT) = (REG_CNTx_LOADVAL) and

MSG_CNTx_SFKT **= 1**

if

CNTx_GENERAL_DISABLE = 0
and

CNTx_SFKT_DISABLE = 0

and

Zx **0** → **1**

**Note**

A software latch retrigger is possible as well (→ see also Software (SW) latch retrigger (page 4-5)). Please use byte 0 (CNT1) or byte 1 (CNT2, bit 5 CNTx_SW_LR
Process output/ control interface (page 3-19).

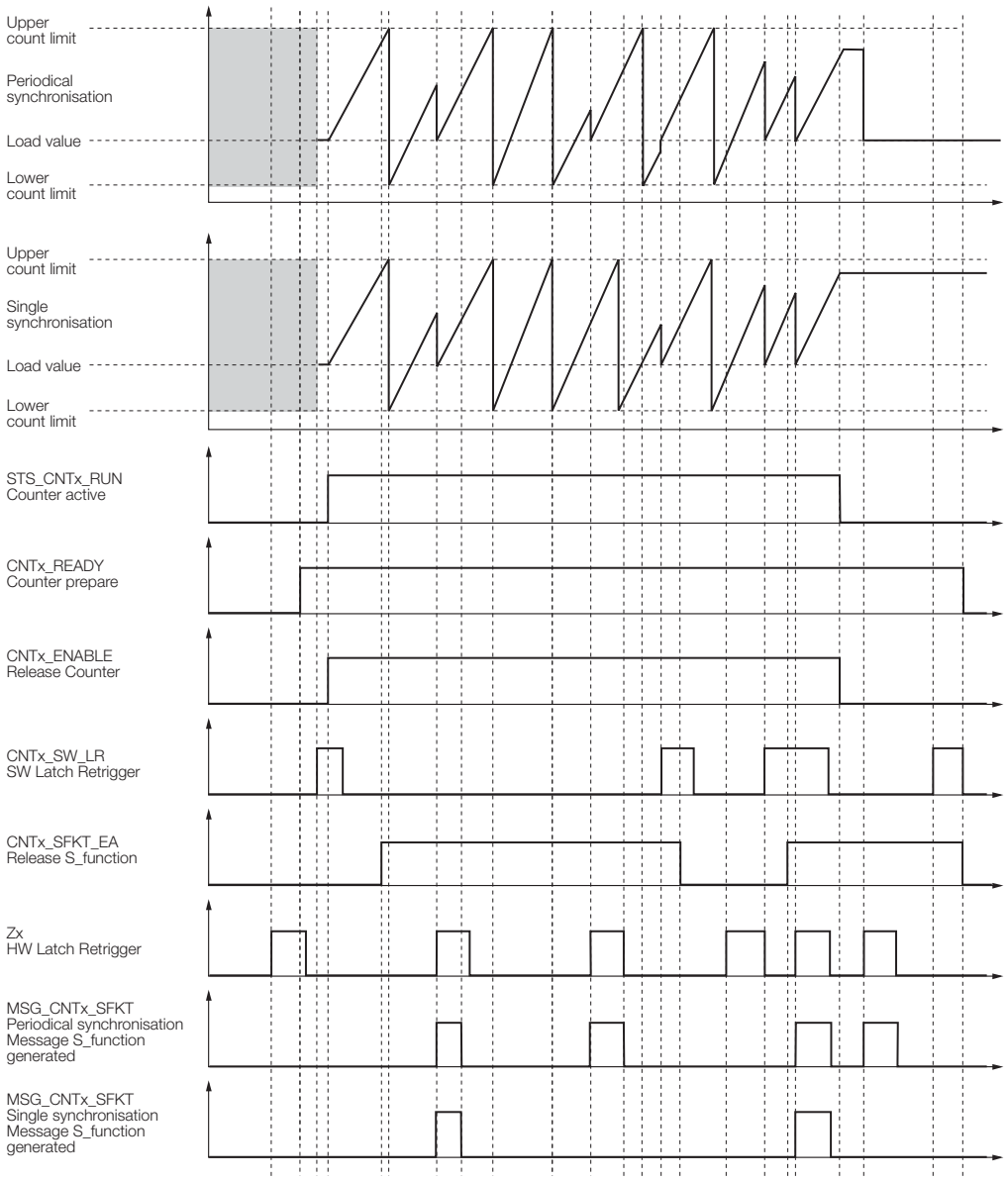
When executing a SW latch retrigger, the following applies:
(REG_CNTx_LATCH) = (REG_CNTx_CNT) and
(REG_CNTx_CNT) = (REG_CNTx_LOADVAL) und
MSG_CNTx_SW_LR **= 1**

if

CNTx_GENERAL_DISABLE = 0
and

CNTx_SW_LR **0** → **1**

*Figure 4-6: synchronization*



## 4.2 Additional functions of the count inputs

### 4.2.1 Additional function: Measurement mode

In addition to the counter function, the counters are capable of carrying out measurements (Frequency measurement (page 4-19) or Period duration measurement (page 4-20)) simultaneously.

The following registers are used to support this function (→ see also Register interface (page 9-4)).

| | Register name | Register no. | Default value | Access |
|---|---|---|---|---|
| Counter 1 | REG_CNT1_MV<br>Measurement value | 33<br>(0×21) | | RO |
| | REG_CNT1_INTTIME<br>Integration time CNTx 10ms/bit<br>**max. 17800** x 10ms) | 41<br>(0×29) | 10<br>0×00 00 00 0A<br>(100 ms) | RW |
| | REG_CNT1_MUL<br>Multiplicator CNTx | 42<br>(0×2A) | 1<br>(0×00 00 00 01) | RW |
| | REG_CNT1_DIV<br>Divisor CNTx | 43<br>(0×2B) | 1<br>(0×00 00 00 01) | RW |
| | REG_CNT1_IPI<br>Counted pulses per integration time | 44<br>(0×2C) | 0<br>(0×00 00 00 00) | RO |
| | REG_CNT1_TO<br>Time-out CNTx, 10ms/ Bit | 35<br>(0×2D) | 0<br>(0×00 00 00 00) | RW |
| Counter 2 | REG_CNT2_MV<br>Measurement value | 65<br>(0×41) | | RO |
| | REG_CNT2_INTTIME<br>Integration time CNTx 10ms/bit<br>**max. 17800** x 10ms) | 73<br>(0×49) | 10<br>0×00 00 00 0A<br>(100 ms) | RW |
| | REG_CNT2_MUL<br>Multiplicator CNTx | 74<br>(0×4A) | 1<br>(0×00 00 00 01) | RW |
| | REG_CNT2_DIV<br>Divisor CNTx | 75<br>(0×4B) | 1<br>(0×00 00 00 01) | RW |
| | REG_CNT2_IPI<br>Counted pulses per integration time | 76<br>(0×4C) | 0<br>(0×00 00 00 00) | RO |
| | REG_CNT2_TO<br>Time-out CNTx, 10ms/ Bit | 77<br>(0×4D) | 0<br>(0×00 00 00 00) | RW |

### Frequency measurement

In frequency measurement mode, the number of count pulses (content of REG_CNT1_IPI (page 9-7)) within an integration time, which has to be defined (REG_CNT1_INTTIME (page 9-7)) gemessen.. This integration time is parameterizable in steps of 10ms/ bit. An integration time of max. 178 s is possible (a min. of 100 ms is reasonable).

→ After the integration time has expired, the result is calculated and entered into the register interface.

**Note**

Changes in the count direction within the integration time cause errors in the frequency measurement.

### Activation of the frequency measurement

The frequency measurement is activated, if:

CNTx_FQPD = 0
(→ see Parameter data of the module, byte 0, bit2, page 3-10).

The frequency measurement is done in the following basic measurement modes:

- Calculation of the frequency in **mHz**:

| Input registers | Value |
|---|---|
| REG_CNT1_MUL | 1 |
| REG_CNT1_DIV | 1 |
| REG_CNT1_INTTIME | Integration time: Multiple of **10 ms** |
| **Output registers** | |
| REG_CNT1_MV | Frequency in **mHz** |

- Calculation of the frequency in **Hz**:

| Input registers | Value |
|---|---|
| REG_CNT1_MUL | 1 |
| REG_CNT1_DIV | 1000 |
| REG_CNT1_INTTIME | Integration time: Multiple of **10 ms** |
| **Output registers** | |
| REG_CNT1_MV | Frequency in **Hz** |

**Note**

The accuracy of the measurement increases with the length of the integration time.

It should be noted, that the measured value (REG_CNTx_MV) is only updated after the integration time has expired.

### Restrictions

The following settings are **not** allowed:
REG_CNTx_MUL = 0,
REG_CNTx_DIV = 0,
REG_CNTx_INTTIME = 0.

### Checking the counter function

In order to check the counter function, a message MSG_CNTx_FQE = 1 can be generated
(→ see page 3-16)

- **if,**
  for a defined Time-out-time in REG_CNTx_TO,
  the content of the register for the measured value REG_CNTx_MV = 0,
- or **if**
  the measured value in REG_CNTx_MV is 0×FF FF FF FF.

If the time-out-time in REG_CNTx_TO = 00 00 00 00 (default-setting), then the message via MSG_CNTx_FQE is switched off.

**Note**

If

REG_CNTx_DIV = 0,
or REG_CNTx_INTTIME = 0
or REG_CNTx_INTTIME > 17800
THEN the value is transferred and an error bit is set in REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

The calculation of the measurement value will then be stopped and REG_CNTx_MV will be set to "0".

### Period duration measurement

It might be suitable to choose the period duration measurement for signal changes with lower frequencies.

### Enabling the period duration measurement

The period duration measurement is activated, if:

CNTx_FQPD = 1
(→ see Parameter data of the module, byte 0, bit2, page 3-10).

The period duration measurement is done in the following basic measurement modes:

- Calculation of the period duration measurement in **µs**:

| Input registers | Value |
|---|---|
| REG_CNT1_MUL | 1 |
| REG_CNT1_DIV | 1 |
| **Output registers** | |
| REG_CNT1_MV | period duration measurement in **µs** |

- Calculation of the period duration measurement in **?s**:

| Input registers | Value |
|---|---|
| REG_CNT1_MUL | 1 |
| REG_CNT1_DIV | 1000 |
| **Output registers** | |
| REG_CNT1_MV | period duration measurement in **ms** |

**Note**

The accuracy of the measurement increases with the length of the integration time.

It should be noted, that the measured value (REG_CNTx_MV) is only updated after the integration time has expired.

### Restrictions

The following settings are **not** allowed:
REG_CNTx_MUL = 0,
REG_CNTx_DIV = 0,
REG_CNTx_INTTIME = 0.

### Checking the counter function

In order to check the counter function, a message MSG_CNTx_FQE = 1 can be generated (→ see page 3-16)

- **if,**
  after a defined time-out-time in REG_CNTx_TO,
  the period duration measurement has not been finished, yet,
- or **if**
  the measured value in REG_CNTx_MV > 0×FF FF FF FF.

If the time-out-time in REG_CNTx_TO = 00 00 00 00 (default-setting), then the message generation in MSG_CNTx_FQE is switched off.

**Note**

IF

REG_CNTx_DIV = 0,
THEN the value is transferred and an error bit is set.

The calculation of measurement values is stopped and REG_CNTx_MV is set to "0".

### Revolutions speed measurement

The revolutions measurement is not executed directly.

In the operation mode "frequency measurement" (→ see parameter measurement mode CNT1), the no. of revolutions (n) in 1/min can be calculated on the basis of the frequency (f).

REG_CNTx_MUL is used to change the time base (for example from s to min) and REG_CNTx_DIV is used to specify the encoder pulses per revolution.

- Revolutions speed in **1/ min**:

| **Input registers** | **Value** |
|---|---|
| REG_CNT1_MUL | 60 |
| REG_CNT1_DIV | Pulses per encoder revolution ×1000 |
| REG_CNT1_INTTIME | Integration time: Multiple of **10 ms** |
| **Output registers** | |
| REG_CNT1_MV | Revolutions speed in **1/ min**: |

- Revolutions speed in **1/ 1000 min**:

| **Input registers** | **Value** |
|---|---|
| REG_CNT1_MUL | 60 |
| REG_CNT1_DIV | Pulses per revolution of the encoder |
| REG_CNT1_INTTIME | Integration time: Multiple of **10 ms** |
| **Output registers** | |
| REG_CNT1_MV | Revolutions speed in **1/ 1000 min**: |

- Revolutions speed in **1/ s**:

| **Input registers** | **Value** |
|---|---|
| REG_CNT1_MUL | 1 |
| REG_CNT1_DIV | Pulses per encoder revolution ×1000 |
| REG_CNT1_INTTIME | Integration time: Multiple of **10 ms** |
| **Output registers** | |
| REG_CNT1_MV | Revolutions speed in **1/s** |

**Note**

The accuracy of the measurement increases with the length of the integration time.

It should be noted, that the measured value (REG_CNTx_MV) is only updated after the integration time has expired.

### Downtime-monitoring

In revolutions speed measurement, downtime-monitoring can be realized by means of a time-out time.

This time is defined as a multiple of 10 ms in REG_CNTx_TO (page 9-7).

If, during this time-out-time

REG_CNTx_MV = 0 ,which means no pulse was detected,

then

MSG_CNTx_FQE = 1 = downtime!

# 5 Functions of the PWM outputs (PWM1 and PWM2)

## 5.1 Functions of the PWM outputs (PWM1 and PWM2)

The outputs P1/ D1 and P2/ D2 provide the two PWM channels. The outputs Px are used for frequency output. The logic status of the outputs Dx can define the direction.

The outputs Px serve to give out a square wave signal with a defined mark-to-space ratio, a defined period duration and a defined number of pulses. Depending on the operation mode, the content of certain registers is used to define the nature of the output signal.

In order to support the PWM function, each channel provides an output Dx, which, for example, can be used as direction signal (→ see Description of the function outputs D1 and D2 (page 6-1)).

Additionally, further functions are provided with input Z1 for PWM1 and input Z2 for PWM2 (→ see Special function of inputs Z1 and Z2 for the PWM (page 5-11)).

### 5.1.1 Module restart with saved values

In case of a power reset, the volatile contents of the PWM-registers (see below) are automatically loaded with the "start values after reset" from the reset-value-registers.

*Table 5-1: Reset-value registers for a restart*

| | |
|---|---|
| REG_PWMx_PD | REG_PWMx_PD_RV |
| REG_PWMx_DC | REG_PWMx_DC_RV |
| REG_PWMx_DHIGH | REG_PWMx_DHIGH_RV |
| REG_PWMx_DLOW | REG_PWMx_DLOW_RV |

### 5.1.2 Operation modes of the PWM outputs Px

The outputs P1 and P2 work as pulse-width modulated outputs.

A square signal with a defined form and of a number of pulses can be given out at these PWM outputs. The function of the output can be chosen as follows via the parameter "mode Px" (→ see Parameter data of the module (page 3-10))

*Table 5-2: Mode PWMx*

| **Bit 3 ... Bit 0 (Value)** | **mode PWM1** (→ see byte 7 of the Parameter data of the module (page 3-10)) | **Mode PWM 2** (→ see byte 9 of the Parameter data of the module (page 3-10)) |
|---|---|---|
| 0000 **A** | **PD DC Definition** (Period Duration / Duty Cycle Definition) (→ see page 5-3) | |
| 0001 | **HT LT Definition** (High Time / Low Time Definition) (→ see page 5-5) | |
| 0010 to 1110 | not defined | |
| 1111 | **P just output**<br>simple digital output, controlled via the Process output/ control interface (page 3-19) | |

**Note**

The parameterization of undefined functions is reported through a diagnostic message with PWMx_PAR_ERR = 1 (→ see Diagnostic data of the module (page 3-9)). IF PWMx_PAR_ERR = 1, THEN the PWM can not be activated.

Additionally to the diagnostic message, an error message is reported in the REG_CONFIG_ERRSTS (PWMx) (page 7-4) when a parameter error was detected.

### Period Duration / Duty Cycle Definition

This operation mode allows:

- Pulse width modulation, → see page 5-4
- Frequency modulation, → see page 5-4

**Registers to be written:**

| Register name | Register no. | Default value | Description |
|---|---|---|---|
| **PWM1** | | | |
| REG_PWM1_PD<br>period duration PWM1 in 41,667 ns/ bit | 96<br>(0×60) | Content of REG_PWM1_PD_RV | volatile, for changes during normal operation |
| REG_PWM1_PD_RV | 104<br>(0×68) | 0 × 00 00 5D C0<br>(= 1000 Hz) | non-volatile, for defined values during start/ reset |
| REG_PWM1_DC<br>mark-to-space ratio PWM1 | 97<br>(0×61) | Content of REG_PWM1_DC_RV | volatile, for changes during normal operation |
| REG_PWM1_DC_RV | 105<br>(0×69) | 0×7F FF FF FF<br>(= 50 %) | non-volatile, for defined values during start/ reset |
| REG_PWM1_CNTSV<br>Load value of the pulses to be given out | 100<br>(0× 64) | 0 × 00 00 27 10<br>(1000 pulses) | non-volatile |
| **PWM2** | | | |
| REG_PWM2_PD<br>Period duration PWM2 in 41,667 ns/ bit | 112<br>(0×70) | Content of REG_PWM2_PD_RV | volatile, for changes during normal operation |
| REG_PWM2_PD_RV | 120<br>(0×78) | 0 × 00 00 5D C0<br>(= 1000 Hz) | non-volatile, for defined values during start/ reset |
| REG_PWM2_DC<br>mark-to-space ratio PWM2 | 113<br>(0×71) | Content of REG_PWM2_DC_RV | volatile, for changes during normal operation |
| REG_PWM2_DC_RV | 121<br>(0×79) | 0×7F FF FF FF<br>(= 50 %) | non-volatile, for defined values during start/ reset |
| REG_PWM2_CNTSV<br>Load value of the pulses to be given out | 116<br>(0× 74) | 0 × 00 00 27 10<br>(1000 pulses) | non-volatile |

**1 Pulse width modulation (PWM**):
Dynamic changing of the pulse width with a constant period duration.

– Period duration (constant):
REG_PWMx_PD (in 41,6667 ns / bit)

– Pulse width (dynamic):
REG_PWMx_DC
The pulse width is the ratio of pulse duration and period duration.
pulse width:
100 % = 0 × FF FF FF FE, corresponds to static ON
50 % = 0 × 7F FF FF FF
0 % = 0 × 00 00 00 00, corresponds to static OFF.

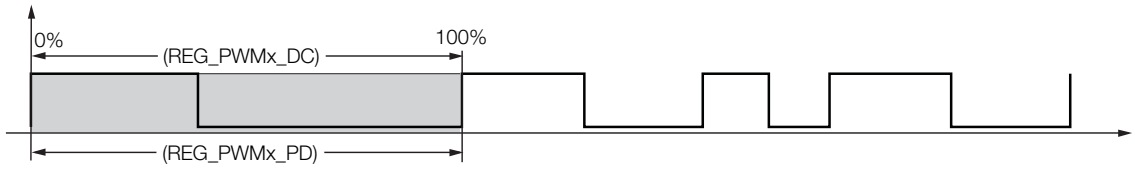
*Figure 5-1: pulse width modulation*


**2 Frequency modulation (FM):**
Changing the frequency for the pulse output by means of a dynamic change of the period duration at a constant mark-to-space-ratio.

– Period duration (dynamic):
REG_PWMx_PD (in 41,6667 ns / bit)
By means of this, the pulse output can be set within the range from 0,005588 Hz to 20 000 Hz.

– Pulse width (constant):
REG_PWMx_DC
The pulse width is the ratio of pulse duration and period duration.

*Figure 5-2: Frequency modulation*



### Application example

Frequency modulation:

Necessary settings:

Parameterization: Operation mode: PD DC Definition

process data: PWMx_SINGLE =1

To give out a signal with 100 Hz and a duty cycle of 50 % for 25000 signal sequences, the following settings have to be done:

*Table 5-3: Reset-value-registers for a restart*

| | |
|---|---|
| REG_PWMx_DC | 0 x 7F FF FF FF (Duty Cycle 50 %) |
| REG_PWMx_PD<br>in steps of 41,667 ns | 0 x 00 03 A9 80 (240000)<br><br>Calculation:<br>100 Hz ≙ period duration = 0,01 s<br>0,01 s/ 41,667 ns ≙<br>$(10 \times 10^{-3})$ s/ $(41{,}667 \times 10^{-9})$ s = 240000 |
| REG_PWMx_CNTSV | 0 × 00 00 61 A8 (25000 signals) |

**Note**

In case of an incorrect setting of the period duration and/ or the pulse width, the change of the register content is transferred and an error message is set in the REG_CONFIG_ ERRSTS (PWMx) (page 7-4).

### High Time / Low Time Definition

In the operation mode "High Time / Low Time Definition", the mark-to-space-ratio of the signal to be given out can exactly be defined through the direct presetting of pulse and space duration.

The content of register REG_PWMx_DHIGH corresponds to the pulse duration, the content of register REG_PWMx_DLOW corresponds to the space duration. Both are set in 41,667ns/ bit. These values can be written directly.

**Note**

The contents of the registers for pulse and space duration (REG_PWMx_DHIGH and REG_PWMx_DLOW) are monitored.
If a pulse or space duration of less than 25 µs is set or if the sum of pulse and space duration (REG_PWMx_DHIGH + REG_PWMx_DLOW) results in a value 0 x FF FF FF FE, then the change of the register content is transferred and an error message is set in the REG_CONFIG_ ERRSTS (PWMx) (page 7-4).

**Registers to be written:**

| Register name | Register no. | Default value | Description |
|---|---|---|---|
| **PWM1** | | | |
| REG_PWM1_DHIGH<br>Pulse duration PWM1 in 41,667 ns/ bit | 98<br>(0× 62) | Content of REG_PWM1_DHIGH_RV | volatile, for changes during normal operation |
| REG_PWM1_DHIGH_RV | 106<br>(0 × 6A) | 0 x 00 00 2E E0<br>(= 500 µs) | non-volatile, for defined values during start/ reset |
| REG_PWM1_CNTSV<br>Load value of the pulses to be given out | 100<br>(0× 64) | 0 × 00 00 27 10<br>(1000 pulses) | non-volatile |
| REG_PWM1_DLOW<br>Space duration PWM1 in 41,667 ns/ bit | 115<br>(0×73) | Content of REG_PWM1_DLOW_RV | volatile, for changes during normal operation |
| REG_PWM1_DLOW_RV | 107<br>(0×6B) | 0 x 00 00 2E E0<br>(= 500 µs) | non-volatile, for defined values during start/ reset |
| **PWM2** | | | |
| REG_PWM2_CNTSV<br>Load value of the pulses to be given out | 116<br>(0× 74) | 0 × 00 00 27 10<br>(1000 pulses) | non-volatile |
| REG_PWM2_DHIGH<br>Pulse duration PWM2 in 41,667 ns/ bit | 98<br>(0× 62) | Content of REG_PWM2_DHIGH_RV | volatile, for changes during normal operation |
| REG_PWM2_DHIGH_RV | 106<br>(0 × 6A) | 0 x 00 00 2E E0<br>(= 500 µs) | non-volatile, for defined values during start/ reset |
| REG_PWM2_DLOW<br>Space duration PWM2 in 41,667 ns/ bit | 115<br>(0×73) | Content of REG_PWM2_DLOW_RV | volatile, for changes during normal operation |
| REG_PWM2_DLOW_RV | 107<br>(0×6B) | 0 x 00 00 2E E0<br>(= 500 µs) | non-volatile, for defined values during start/ reset |

### 5.1.3 Continuous signal output

Process output/ control interface (page 3-19): **PWMx_SINGLE = 0**

The signal output is continuous. The signal form is changeable during the signal output by changing the corresponding register entries.

The signal output can be started or stopped using the SW gate (**or**, if parameterized, using the HW gate,→ see also Enabling the pulse output (page 5-9)). The value of the 32 bit counters is preserved.

If PWMx_GENERAL_DISABLE is set to "1", the PWM functions are disabled. The register contents are preserved.

Setting PWMx_GENERAL_DISABLE to "0" generally re-enables the PWM. It continues to work with existing register contents.

Existing messages (MSG) should be reset when setting the PWM (→ see chapter 8, Error handling in the control interface/ check-back interface (page 8-1)).

**Procedure of the continuous signal output:**

| | | |
|---|---|---|
| **1** | Set register contents to define the signal form: | |
| **1.1** | REG_PWMx_CNTSV | Number of pulses to be given out |
| **1.2** | REG_PWMx_DC, REG_PWMx_PD<br>or | Operation mode: Period Duration / Duty Cycle Definition (page 5-3) |
| **1.3** | REG_PWMx_DLOW,<br>REG_PWMx_DHIGH | Operation mode: High Time / Low Time Definition (page 5-5) |
| **2** | PWMx_SW_LR 0 → 1 | The latch retrigger causes that the start value is copied from REG_PWMx_CNTSV into REG_PWMx_CNTDC. |
| **3** | REG_PWMx_CNTSV<br>→ REG_PWMx_CNTDC | |
| **4** | PWMx_ENABLE 0 → 1 | Setting the enable-bit, the signal output begins |
| **5** | REG_PWMx_CNTDC | The count value in the register is decremented with every signal change 1 → 0 of the output signal<br>REG_PWMx_CNTDC = 0. |
| **6** | REG_PWMx_CNTDC = 0. | Number of pulses to be given out<br>REG_PWMx_CNTDC = 0. |
| **7** | MSG_PWMx_NDDC = 1 | Message *„zero crossing occurred"* (→ see Process input/ check-back interface (page 3-14)). |
| **8** | PWMx_LOGMSG 1 → 0 | The message has to be reset!<br>To do so, PWMx_LOGMSG is first set from 0→ 1 and then from 1→ 0 (→ see Process output/ control interface (page 3-19)) or Storage of messages (MSG) (page 8-4)). |
| **9** | REG_PWMx_LATCH =<br>REG_PWMx_CNTDC<br>→ REG_PWMx_LATCH = 0 | Number of pulses to be given out = 0.<br>The value is now copied to the latch register, this register is = 0, too.<br>A "0" in the latch register causes an automatic latch retrigger, → see also Special function Zx (PWM): Hardware latch retrigger (page 5-14). |
| **10** | REG_PWMx_CNTDC =<br>REG_PWMx_CNTSV | The start value from REG_PWMx_CNTSV is reloaded into the register of pulses to be given out.<br>→ The signal output is continued with the start value. |

### 5.1.4 Periodical signal output

Process output/ control interface (page 3-19): **PWMx_SINGLE = 1**

The signal output is defined as single. The signal form is changeable during the signal output by changing the corresponding register entries.

The signal output can be started or stopped using the SW gate (or, if parameterized, using the HW gate). The value of the 32 bit counters is preserved.

If PWMx_GENERAL_DISABLE is set to "1", the PWM functions are disabled. The register contents are preserved.

Setting PWMx_GENERAL_DISABLE to "0" generally re-enables the PWM. It continues to work with existing register contents.

Existing messages (MSG) should be reset when setting the PWM (→ see chapter 6, Description of the function outputs D1 and D2 (page 6-1)).

**Procedure of the periodical signal output:**

| | | |
|---|---|---|
| **1** | Set register contents to define the signal form: | |
| **1.4** | REG_PWMx_CNTSV | Number of pulses to be given out |
| **1.5** | REG_PWMx_DC, REG_PWMx_PD or | for operation mode: Period Duration / Duty Cycle Definition (page 5-3) |
| **1.6** | REG_PWMx_DLOW, REG_PWMx_DHIGH | for operation mode: High Time / Low Time Definition (page 5-5) |
| **2** | PWMx_SW_LR 0 → 1 | The latch retrigger causes that the start value is copied from REG_PWMx_CNTSV into REG_PWMx_CNTDC. |
| **3** | REG_PWMx_CNTSV → REG_PWMx_CNTDC | |
| **4** | PWMx_ENABLE 0 → 1 | Setting the enable-bit, the signal output begins STS_PWMx_RUN = 1 |
| **5** | REG_PWMx_CNTDC | The count value in the register is decremented with every signal change 1 → 0 of the output signal until REG_PWMx_CNTDC = 0. |
| **6** | REG_PWMx_CNTDC = 0. | Number of pulses to be given out REG_PWMx_CNTDC = 0. |
| **7** | MSG_PWMx_NDDC = 1 | Message „*zero crossing occurred*" (→ see Process input/ check-back interface (page 3-14)). |
| **8** | PWMx_LOGMSG → 0 | The message has to be reset!<br>To do so, PWMx_LOGMSG is first set from 0 → 1 and then from 1 → 0 (→ see Process output/ control interface (page 3-19)) or Storage of messages (MSG) (page 8-4)). |
| **9** | STS_PWMx_RUN = 0 | The signal output is stopped, because REG_PWMx_CNTDC = 0. |
| **10** | PWMx_SW_LR 0 → 1 | With another latch retrigger the procedure restarts and the signal output restarts, as long as PWMx_ENABLE = 1. |

### 5.1.5 Enabling the pulse output

**Prerequisite**:

Enabling the PWM via hard- or software gate requires a general enabling of the function using PWM1_ GENERAL_ DISABLE = 0 (default-setting).

**Note**

The counter can be enabled either per software **or** per hardware gate:

**NOT** PWM1_ GENERAL_ DISABLE

**AND** (SW-Tor **OR** HW-Tor)

**Hardware gate (HW gate)**

The pulse output is enabled with signal Zx = 1 and disabled with Zx = 0.

In order to use Zx to enable the pulse output, the special function of the output has to be enabled as well (→ see Special function Zx (PWM): HW gate (page 5-12)).

The following applies for the HW gate:

STS_PWMx_RUN = 1, if

PWMx_GENERAL_DISABLE = 0
and

PWMx_SFKT_DISABLE = 0 and

Zx = 1

**Software-gate (SW gate)**

The enabling of the counter is done using a signal change 0→ 1 at bit PWM1_ ENABLE (PWM1) or PWM2_ ENABLE (PWM2) in the Process output/ control interface (page 3-19).

The following always applies for the SW gate:

STS_PWMx_RUN = 1,if

PWMx_GENERAL_DISABLE = 0 and

PWMx_ENABLE = 1

The signal output is done with an open SW gate PWMx_GENERAL_DISABLE = 0 and PWMx_ENABLE = 1 as long as REG_PWMx_CNTDC <> 0.

### 5.1.6 Latch retrigger (PWM)

The number of pulses to be given out is retriggered, the current value of the decrement register REG_PWMx_CNTDC is stored to the latch register ((REG_PWMx_LATCH) = (REG_PWMx_CNTDC)) and the load value from REG_PWMx_CNTSV is loaded again to the decrement register ((REG_PWMx_CNTDC) = (REG_PWMx_CNTSV)). The signal output is continued.

The pulse output has to be enabled (→ see Enabling the pulse output (page 5-9)) in order to execute this latch retrigger.

**Hardware (HW) latch retrigger**

The hardware latch retrigger is done via a signal change at Zx 0 → 1 (→ siehe Special function Zx (PWM): Hardware latch retrigger (page 5-14)).

**Software (SW) latch retrigger**

The software latch retrigger is done by setting the bits PWMx_SW_LR in the Process output/ control interface (page 3-19). The executed SW latch retrigger is confirmed through bit MSG_PWMx_SW_LR of the Process input/ check-back interface (page 3-14) (bit 0 in byte 4 (PWM1) and byte 5 (PWM2)).

### 5.1.7 Special function of inputs Z1 and Z2 for the PWM

The inputs Z1 and Z2 can be used to support the PWM functions (or the counter functions, → see chapter 4).

**Note**

Each special function of Zx is **enabled** per default via PWMx_SFKT_DISABLE **= 0** in the Process output/ control interface (page 3-19). The enabling of this function is acknowledged via STS_PWMx_SFKT_EN = 1.
**The disabling** of the special function is done via PWMx_SFKT_DISABLE **= 1**.

Possible functions of the PWM outputs (PWM1 and PWM2) at a signal change 0→ 1:

- alarm (→ see Special function Zx (PWM): alarm (page 5-12))
- HW gate (enabling the pulse output, → see Special function Zx (PWM): HW gate (page 5-12))
- Synchronization (latch retrigger, single or periodical with the load value, → see Special function Zx (PWM): Hardware latch retrigger (page 5-14))

*Table 5-4: functions Z1 and Z2*

| **Bit 7 ... Bit 4 (Value)** | **Mode Z1** (→ see byte 0 of the Parameter data of the module (page 3-10)) | **Mode Z1** (→ see byte 1 of the Parameter data of the module (page 3-10)) |
|---|---|---|
| 0000 to 0101 | Functions for PWM1 and PWM2, → see Special function of inputs Z1 and Z2 at CNTx (page 4-12). | |
| 0110 | reserved | – |
| 0111 | **Alarm input PWM** | |
| | A signal change 0→ 1 at Zx causes the setting of the MSG_CNTx_SFKT flag. This serves to also report short-time events. | |
| 1000 | **HW gate PWM**<br>The enabling can also be done per software gate(→ see also here Enabling the pulse output (page 5-9)) | |
| | – with Zx = 0, the signal output is disabled<br>– with Zx = 1, the signal output is enabled | |
| | **Retrigger PWM** | |
| 1001 | In single signal output, the counter of the pulses to be given out (REG_PWMx_CNTDC) is reloaded and the signal output is enabled. | |
| 1010 to 1110 | reserved | |
| 1111 | **Z just input**<br>Simple digital input, the status is reported via the check-back interface. | |

### Special function Zx (PWM): alarm

If Zx is parameterized as analarm-signal for the PWM (Parameter data of the module (page 3-10)), a message is generated when the signal is triggered.

→ Enable/disable the special function (→ see page 5-11)

After an alarm-signal occurred, bit MSG_PWMx_SFKT page 3-14 in the Process input/ check-back interface is set.

**Note**

When using alarm-signals as Open Collector, a pull up resistance can be switched via the Parameter data of the module (page 3-10).

### Special function Zx (PWM): HW gate

→ Enable/disable the special function (→ see page 5-11)

If Zx is parameterized as HW gate for the PWM (Parameter data of the module (page 3-10)),
the signal output at the PWM output is

started with Zx = 1
and
stopped with Zx = 0.

The special function of Zx has to be enabled via PWMx_SFKT_DISABLE = 0 in the Process output/ control interface. The enabling of this function is acknowledged via STS_PWMx_SFKT_EN = 1.

After opening the HW gate, bit MSG_PWMx_SFKT page 3-14 is set in the Process input/ check-back interface and can only be reset after the HW gate is closed again.

### Start/ Stop via HW gate

**Note**

In addition to starting and stopping the PWM output using Zx (HW gate), the output can be started or stopped using PWMx_ENABLE (SW gate) (see also Enabling the pulse output (page 5-9)).

Stopping the PWM output interrupts the signal output. The current signal status of the output is preserved during this interruption. After a restart of the output, the signal output is continued with the conditions that were active when the output was stopped.

### Output signal

During the enabling via HW or SW gate, a PWM output signal is generated depending on the register contents of REG_PWMx_DC und REG_PWMx_PD.

In the continuous signal output, decrementing the counter (REG_PWMx_CNTDC) causes a latch retrigger at "0".

This means, the output counter

(REG_PWMx_LATCH) = (REG_PWMx_CNTDC)

and

is set to (REG_PWMx_CNTDC) = (REG_PWMx_CNTSV). The signal output is not interrupted.

MSG_PWMx_NDDC reports the zero crossing, which means the expiration of the counter.

The following applies for the HW gate:

STS_PWMx_RUN = 1,if

PWMx_GENERAL_DISABLE = 0

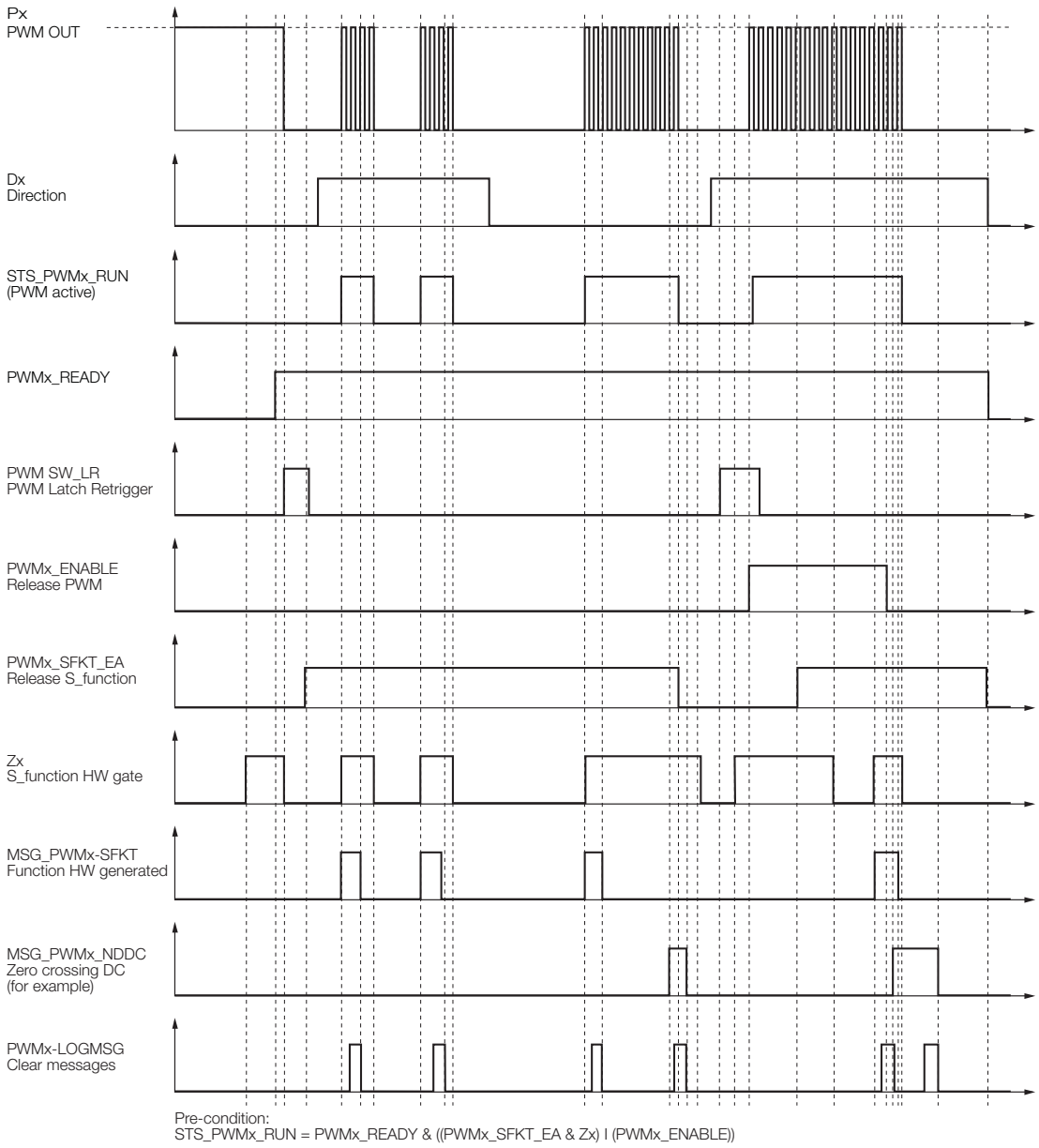
and

PWMx_SFKT_DISABLE = 0

and

Zx = 1

*Figure 5-3: Pulse output with HW and SW gate*



### Special function Zx (PWM): Hardware latch retrigger

If Zx is parameterized as latch retrigger signal for the PWM output (Parameter data of the module (page 3-10)), Zx is used as hardware (HW) latch retrigger

→ Enable/disable the special function (→ see page 5-11)

With a signal change 0→ 1 at input Zx,

**1** The content of register REG_PWMx_CNTDC is stored to REG_PWM_LATCH page 9-9,

**2** and REG_PWMx_CNTDC is reloaded via REG_PWMx_CNTSV .

**3** If the enabling is set via SW gate (PWMx_ENABLE), the signal output is started immediately.

After a latch retrigger event occurred, bit MSG_ PWM1_ SFKT (page 3-14)
MSG_ PWM2_ SFKT (page 3-14) in the check-back interface is set. It can then be reset via bit CNTx_LOGMSG page 3-19 in the control interface with 0 → 1→ 0.

- Single signal output:
  If the single signal output (PWMx_SINGLE = 1) is parameterized in the Process output/ control interface (page 3-19), the function is only executed once with the **first** signal change 0 → 1 at Zx after enabling the output with PWMx_SFKT_DISABLE = 0.
- Periodical signal output:
  If the periodical signal output (PWMx_SINGLE = 0) is parameterized in the Process output/ control interface (page 3-19), the function is executed with the **every** signal change 0 → 1 at Zx after enabling the output with PWMx_SFKT_DISABLE = 0.

When executing a **HW** latch retrigger, the following applies:

(REG_PWMx_LATCH) = (REG_PWMx_CNTDC) and

(REG_PWMx_CNTDC) = (REG_PWMx_CNTSV) and

MSG_PWMx_SFKT = 1, if

PWMx_GENERAL_DISABLE = 0
and

PWMx_SFKT_DISABLE = 0

and

Zx 0 → 1

**Note**

A **software (SW) latch retrigger** is also possible (→ see also Latch retrigger (PWM) (page 5-10)). Please use byte 2 (PWM1) or byte 3 (PWM2) bit 5 PWMx_SW_LR of the Process output/ control interface (page 3-19).

When executing a SW latch retrigger, the following applies:
(REG_PWMx_LATCH) = (REG_CNTx_CNTDC) and
(REG_PWMx_CNTDC) = (REG_CNTx_CNTSV) and
MSG_PWMx_SW_LR = 1

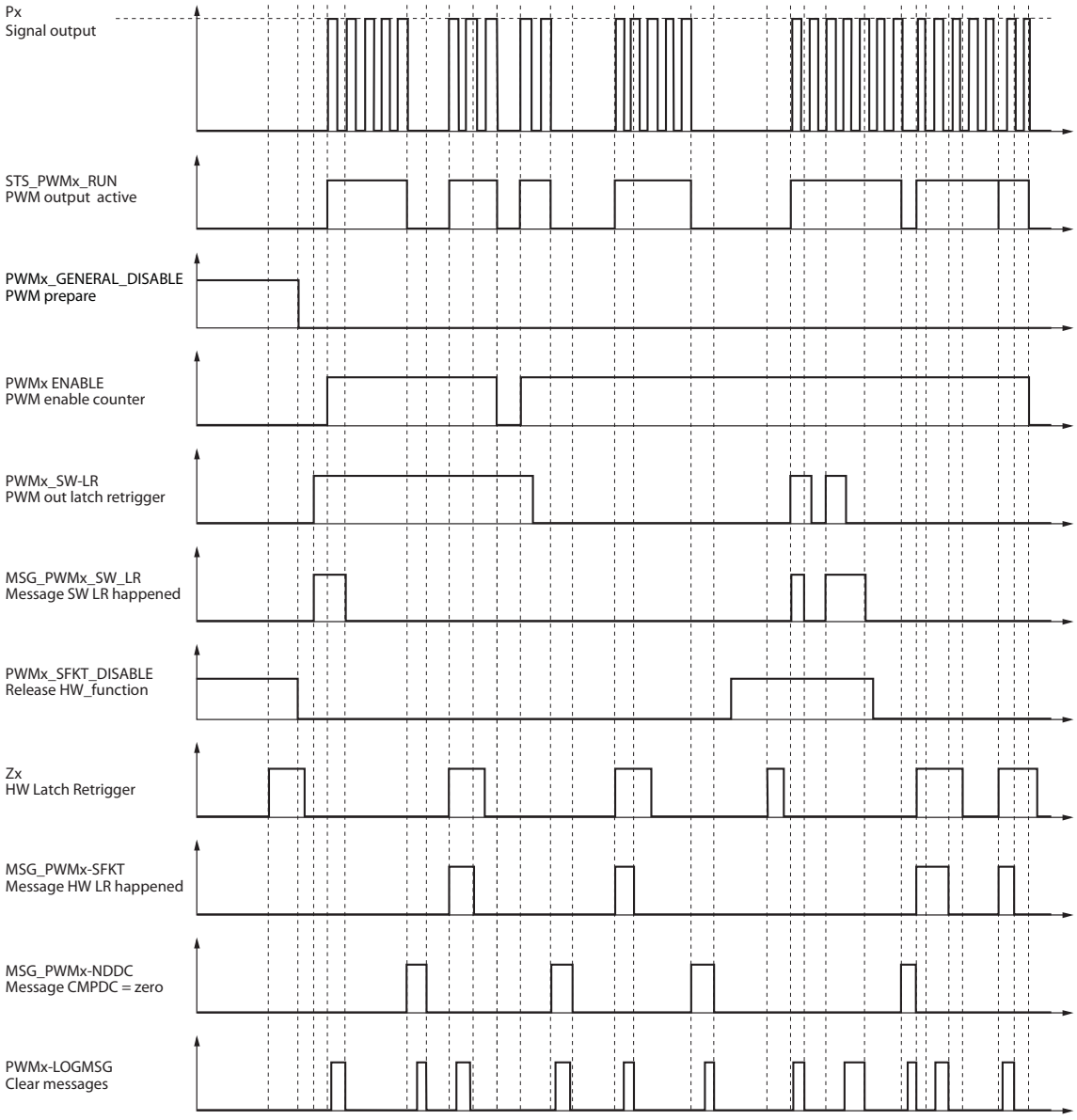
if

PWMx_GENERAL_DISABLE = 0 and

PWMx_SW_LR 0 → 1

With every valid latch retrigger event, the register is always reloaded with the (REG_PWMx_CNTSV) and decremented with every pulse output until it becomes "0".

*Figure 5-4: Latch retrigger at the PWM*



# 6 Description of the function outputs D1 and D2

## 6.1 General

The outputs D1 and D2 can be used multifunctionally. They support both, the function of CNTx as well as the function of PWMx.

Each output can generally be used as simple output (default-parameterization).

### 6.1.1 Direct access to Dx

The output status of D1 and D2 can be changed directly using the bits SET_ D1 and SET_ D2 in the Process output/ control interface (page 3-19). For example, it can be used as direction signal for the PWM.

### 6.1.2 Parameterization of the function "mode Dx"2

By using the parameter "mode Dx" (→ see Parameter data of the module (page 3-10)), other functions can be defined for the outputs.

**Simple functions of the outputs Dx**

*Table 6-1: Simple Functions Dx*

**A** *Default-setting*

| **Bit 5 ... Bit 0 (Value)** | **mode Dx** (→ see byte 6 (D1) and byte 8 (D2) of the Parameter data of the module) |
|---|---|
| 00 0000 | D=STS_CNT_GENERAL_EN |
| 00 0001 | D=STS_CNT_RUN |
| 00 0010 | D=STS_CNT_SFKT_EN |
| 00 0011 | reserved |
| 00 0100 | D=STS_CNT_DIR |
| 00 0101 | D=Z |
| 00 0110 | D=B |
| 00 0111 | D=A |
| 00 1000 | D=MSG_CNT_CMP0 |
| 00 1001 | D=MSG_CNT_CMP1 |
| 00 1010 | D=MSG_CNT_UFLW |
| 00 1011 | D=MSG_CNT_OFLW |
| 00 1100 | D=MSG_CNT_ND |
| 00 1101 | D=MSG_CNT_FQE |
| 00 1110 | D=MSG_CNT_SFKT |
| 00 1111 | D=MSG_CNT_SW_LR |
| 01 0000 | D=MSG_PWM_SW_LR |
| 01 0001 | D=MSG_PWM_NDDC |
| 01 0010 | D=MSG_PWM_SFKT |

*Table 6-1: Simple Functions Dx*

| **Bit 5 ... Bit 0 (Value)** | **mode Dx** (→ see byte 6 (D1) and byte 8 (D2) of the Parameter data of the module) |
|---|---|
| 01 0011 | reserved |
| 01 0100 | D=STS_PWM_GENERAL_EN |
| 01 0101 | D=STS_PWM_RUN |
| 01 0110 | D= STS_PWM_SFKT_EN |
| 01 0111 | reserved |
| 01 1000 | D=1 at CNT=0 |
| 01 1001 | D=1 at CMP0<=CNT<=CMP1 |
| | REG_CNT1_CMP 0 ≤ REG_CNT1_CNT ≤ REG_CNT1_CMP |
| 01 1010 | D=1 at UFLW<=CNT<=CMP0 |
| | REG_CNTx_UFLW ≤ REG_CNTx_CNT ≤ REG_CNTx_CMP0 |
| 01 1011 | D=1 at CMP1<=CNT<=OFLW |
| | REG_CNT1_CMP1 ≤ REG_CNT1_CNT ≤ REG_CNT1_OFLW |
| 01 1100 | reserved |
| 01 1101 | |
| 01 1111 | |
| 11 1111 **A** | D1 = simple output to be controlled via the process data |

**A** *Default-setting*

### Special functions of the outputs Dx: Pulse output defined by time

The outputs can be used to give out a pulse in case of an occurring event defined via the parameter "mode Dx".

If the parameterized event occurs, Dx is switched on for a defined time (pulse duration).

The pulse duration can be set in the following registers in the Register interface (page 9-4) with a resolution of 10 ms/ bit.

| Register name | Register no. | Default value |
|---|---|---|
| REG_CNT1_DO1_IMP<br>Pulse time for a pulse output at D1 in 10 ms/bit | 48<br>(0×30) | 10 = 100 ms<br>(0×00 00 00 A0) |
| REG_CNT2_DO2_IMP<br>Pulse time for a pulse output at D2 in 10 ms/bit | 80<br>(0×50) | |

*Table 6-2: Special functions Dx, pulse output*

| **Bit 5 ... Bit 0 (Value)** | **mode Dx** (→ see byte 6 (D1) and byte 8 (D2) of the Parameter data of the module (page 3-10)) |
|---|---|
| 10 0000 | D=1 for Tx at MSG_CNT_CMP0 |
| | Dx 0 → 1 at MSG_CNTx_CMP0 0 → 1<br>Dx is switched on for defined pulse time if the count value is equal to the compare value 0 (MSG_CNTx_CMP0 (page 8-2)). |
| 10 0001 | D=1 for Tx at MSG_CNT_CMP1 |
| | Dx 0 → 1 at MSG_CNTx_CMP1 0 → 1<br>Dx is switched on for defined pulse time if the count value is equal to the compare value 1 (MSG_CNTx_CMP1 (page 8-2)). |
| 10 0010 | D=1 for Tx at MSG_CNT_UFLW |
| | Dx 0 → 1 at MSG_CNTx_UFLW 0 → 1<br>Dx is switched on for defined pulse time if an underflow of the count value has been detected (MSG_CNTx_UFLW (page 8-2)). |
| 10 0011 | D=1 for Tx at MSG_CNT_OFLW |
| | Dx 0 → 1 at MSG_CNTx_OFLW 0 → 1<br>Dx is switched on for defined pulse time if an overflow of the count value has been detected (MSG_CNTx_UFLW (page 8-2)). |
| 10 0100 | D=1 for Tx at MSG_CNT_ND |
| | Dx 0 → 1 at MSG_CNTx_ND 0 → 1<br>Dx is switched on for defined pulse time if a zero crossing of the count value has been detected (MSG_CNTx_ND (page 8-2)). |
| 10 0101 | D=1 for Tx at MSG_CNT_FQE |
| | Dx 0 → 1 at MSG_CNTx_FQE 0 → 1<br>Dx is switched on for defined pulse time if, within a defined time, no counter pulse has been received(MSG_CNTx_FQE (page 8-2)). |
| 10 0110 | D=1 for Tx at MSG_CNT_SFKT |
| | Dx 0 → 1 at MSG_CNTx_SFKT 0 → 1<br>Dx is switched on for defined pulse time if an event according to the parameterized special function occurred (MSG_CNTx_SFT (page 8-2)). |
| 10 0111 | D=1 for Tx at MSG_CNT_ND |
| | Dx 0 → 1 at MSG_CNTx_SW_LR 0 → 1<br>Dx is switched on for defined pulse time if a software latch retrigger at CNTx was executed (MSG_CNTx_SW_LR (page 8-2)). |
| 10 1000 | D=1 for Tx at MSG_PWM_SW_LR |
| | Dx 0 → 1 at MSG_PWMx_SW_LR 0 → 1<br>Dx is switched on for defined pulse time if a software latch retrigger at PWMx was executed (MSG_PWMx_SW_LR (page 8-3)). |

*Table 6-2: Special functions Dx, pulse output*

| **Bit 5 ... Bit 0 (Value)** | **mode Dx** (→ see byte 6 (D1) and byte 8 (D2) of the Parameter data of the module (page 3-10)) |
|---|---|
| 10 1001 | D=1 for Tx at MSG_PWM_NDDC |
| | Dx 0 → 1 at MSG_PWMx_NDDC 0 → 1<br>Dx is switched on for defined pulse time if a zero crossing has been detected at the PWM (MSG_PWMx_NDDC (page 8-3)). |
| 10 1010 | D=1 for Tx at MSG_PWM_SFKT |
| | Dx 0 → 1 at MSG_PWMx_SFKT 0 → 1<br>Dx is switched on for defined pulse time if an event according to the parameterized special function occurred (MSG_PWMx_SFKT (page 8-3)). |
| 10 1011 | reserved |
| 10 1100 | D=1 for Tx at MSG_CNT_CMP0 OR 1 |
| | Dx 0 → 1 at MSG_CNTx_CMP**0** 0 → 1 or<br>Dx 0 → 1 at MSG_CNTx_CMP**1** 0 → 1<br>Dx is switched on for a defined pulse time if the counter is equal to the compare value **0** *or* to the compare value **1** (MSG_CNTx_CMP0 or MSG_CNTx_CMP1 (page 8-2)). |

**Special functions of the outputs Dx: Pulse output defined by a hysteresis**

The output Dx is switched on due to an event parameterized via the parameter "mode Dx" and only switched off again if this event value has been changed about the hysteresis value.

This avoids that the output is permanently switched on and off in case of a count value toggling around the parameterized switching event (example, → see page 6-6).

The hysteresis can be set as a number of pulses in the following registers within the register interface:

| **Register name** | **Register no.** | **Default value** |
|---|---|---|
| REG_CNT1_DO1_HYS<br>Hysteresis of the D1 for CNT1 | 47<br>(0×2F0) | 10 = 10 pulses<br>(0×00 00 00 0A) |
| REG_CNT2_DO1_HYS<br>Hysteresis of the D2 for CNT2 | 79<br>(0×4F0) | |

*Table 6-3: Special functions Dx, hysteresis*

| **Bit 5 ... Bit 0 (Value)** | **mode Dx** (→ see byte 6 (D1) and byte 8 (D2) of the Parameter data of the module (page 3-10)) |
|---|---|
| 11 0000 | D=1 at CNT< CMP0 Hys. |
| | Dx 0 → 1 at REG_CNTx_CNT < (REG_CNTx_CMP0 - REG_CNTx_DO1_HYS)<br>output 0 → 1, if<br>count value < (compare value 0 - hysteresis value)<br>output 1 → 0, if<br>count value ≥ compare value 0 |

*Table 6-3: Special functions Dx, hysteresis*

| **Bit 5 ... Bit 0 (Value)** | **mode Dx** (→ see byte 6 (D1) and byte 8 (D2) of the Parameter data of the module (page 3-10)) |
|---|---|
| 11 0001 | D=1 at CMP0 < CNT < CMP1 Hys. |
| | Dx 0 → 1 at REG_CNTx_CNT < (REG_CNTx_CMP1 **-** REG_CNT1_DO1_HYS)<br>***AND*** REG_CNT1_CNT > (REG_CNT1_CMP0 **+** REG_CNT1_DO1_HYS)<br>output 0 → 1, if<br>count value < (compare value 1 **-**hysteresis value) *and if*<br>count value > (compare value 0 **+** hysteresis value)<br>output 1 → 0, if<br>count value ≤ compare value 1 or compare value 0 |
| 11 0010 | D=1 at CNT > CMP1 Hys |
| | Dx 0 → 1 at REG_CNTx_CNT >(REG_CNTx_CMP1+ REG_CNTx_DO1_HYS)<br>output 0 → 1, if<br>count value > compare value 1 **+** hysteresis value<br>output 1 → 0, if<br>count value ≤ compare value 1 |
| 11 0011 | D=1 at CNT > CMP1 OR < CMP0 Hys. |
| | Dx 0 → REG_CNT1_CNT < (REG_CNT1_CMP0 **-** REG_CNT1_DO1_HYS)<br>***OR*** REG_CNT1_CNT > (REG_CNT1_CMP1 **+** REG_CNT1_DO1_HYS)<br>output 0 → 1, if<br>count value < (compare value 0 **-**hysteresis value) *and if*<br>count value > (compare value 1 **+** hysteresis value)<br>output 1 → 0, if<br>count value ≥ compare value 0 or ≤ compare value 1 |

Example:

**hysteresis:**

10 pulses (REG_CNT1_DO1_HYS = 10)

**switching event:**

Mode Dx
D=1 at CNT< CMP0 Hys.,

this means:
output 0 → 1, if count value (REG_CNTx_CNT) compare value 0 (REG_CNTx_CMP0)

If the count value toggles around the compare value, the output is **only** switched when the count value differs by more then **-** 10 pulses from the compare value.

# 7 Report of configuration errors

## 7.1 The error-register

The register REG_CONFIG_ERRSTS (register no. 0×0A) in the Register interface (page 9-4) is used to report configuration errors.

If REG_CONFIG_ERRSTS ≠ 0, then bit STS_CONFIG_ERR = 1 in the Process input/ check-back interface (page 3-14) reports that errors in the device's configuration occurred.

It can be read from the register, which error is existent.

### 7.1.1 Error messages in REG_CONFIG_ERRSTS for the count operation mode (CNT1 and CNT2)

*Table 7-1: REG_CONFIG_ERRSTS (CNTx)*

| | Bit | Meaning<br>at bit = 1 |
|---|---|---|
| CNT 1 | 0 | The count value of CNT1 exceeds the count limits.<br>REG_CNT1_CNT > REG_CNT1_HILIMIT **or** REG_CNT1_CNT < REG_CNT1_LOLIMIT |
| | 1 | The load value of CNT1 exceeds the count limits.<br>REG_CNT1_LOADVAL > REG_CNT1_HILIMIT **or** REG_CNT1_CNT < REG_CNT1_LOADVAL |
| | 2 | The configuration of the count limits is faulty.<br>REG_CNT1_HILIMIT ≤ REG_CNT1_LOLIMIT |
| | 3 | The compare value CMP0 of CNT1 exceeds the count limits.<br>REG_CNT1_CMP0 > REG_CNT1_HILIMIT **or** REG_CNT1_CMP0 < REG_CNT1_LOLIMIT |
| | 4 | The compare value CMP1 of CNT1 exceeds the count limits.<br>REG_CNT1_CMP0 > REG_CNT1_HILIMIT **or** REG_CNT1_CMP0 < REG_CNT1_LOLIMIT |
| | 5 | Division through "0" or multiplication with "0" during the measurement at CNT1<br>REG_CNT1_DIV = 0 **or** REG_CNT1_MUL = 0 |
| | 6 | Only valid for frequency measurement:<br>REG_CNT1_INTTIME = 0 **or** REG_CNT1_INTTIME > 17800 |
| | 7 | A diagnostic message is pending.<br>The diagnostic byte 0 for CNT1 > 0 (→ see Diagnostic data of the module (page 3-9)). |
| CNT2 | 8 | The count value of CNT1 exceeds the count limits.<br>REG_CNT2_CNT > REG_CNT2_HILIMIT **or** REG_CNT2_CNT < REG_CNT2_LOLIMIT |
| | 9 | The load value of CNT1 exceeds the count limits.<br>REG_CNT2_LOADVAL > REG_CNT2_HILIMIT **or** REG_CNT2_CNT < REG_CNT2_LOADVAL |
| | 10 | The configuration of the count limits is faulty.<br>REG_CNT2_HILIMIT ≤ REG_CNT2_LOLIMIT |
| | 11 | The compare value CMP0 of CNT1 exceeds the count limits.<br>REG_CNT2_CMP0 > REG_CNT2_HILIMIT **or** REG_CNT2_CMP0 < REG_CNT2_LOLIMIT |
| | 12 | The compare value CMP1 of CNT1 exceeds the count limits.<br>REG_CNT2_CMP1 > REG_CNT2_HILIMIT **or** REG_CNT2_CMP1 < REG_CNT2_LOLIMIT |
| | 13 | Division through "0" or multiplication with "0" during the measurement at CNT1<br>REG_CNT2_DIV = 0 **or** REG_CNT2_MUL = 0 |
| | 14 | Only valid for frequency measurement:<br>REG_CNT2_INTTIME = 0 **or** REG_CNT2_INTTIME > 17800 |
| | 15 | A diagnostic message is pending.<br>The diagnostic byte 1 for CNT2 > 0 (→ see Diagnostic data of the module (page 3-9)). |

### 7.1.2 Error messages in REG_CONFIG_ERRSTS for the PWM output (PWM1 and PWM2)

*Table 7-2: REG_CONFIG_ERRSTS (PWMx)*

| | Bit | Meaning<br>at bit = 1 |
|---|---|---|
| PWM 1 | 16 | Period Duration / Duty Cycle Definition (page 5-3):<br>Invalid setting of period duration and/ or duty cycle (pulse width)<br>High Time / Low Time Definition (page 5-5)<br>Invalid pulse time set (lower than 22 µs).<br>REG_PWM1_DHIGH < 0 x 00 00 02 21 |
| | 17 | Period Duration / Duty Cycle Definition (page 5-3):<br>Invalid setting of period duration and/ or duty cycle (pulse width)<br>High Time / Low Time Definition (page 5-5)<br>Invalid space time set (lower than 22 µs).<br>REG_PWM1_DLOW < 0 x 00 00 02 21 |
| | 18 | invalid period duration set.<br>REG_PWM1_PD > 0 x FF FF FF FE |
| | 19 | invalid duty cycle (pulse width) set<br>REG_PWM1_DC > 0 x FF FF FF FE |
| | 20 to 22 | reserved |
| | 23 | A diagnostic message is pending.<br>The diagnostic byte 2 for PWM1 > 0 (→ see Diagnostic data of the module (page 3-9)). |
| PWM 2 | 24 | Period Duration / Duty Cycle Definition (page 5-3):<br>Invalid setting of period duration and/ or duty cycle (pulse width)<br>High Time / Low Time Definition (page 5-5)<br>Invalid pulse time set (lower than 22 µs).<br>REG_PWM2_DHIGH < 0 x 00 00 02 21 |
| | 25 | Period Duration / Duty Cycle Definition (page 5-3):<br>Invalid setting of period duration and/ or duty cycle (pulse width)<br>High Time / Low Time Definition (page 5-5)<br>Invalid space time set (lower than 22 µs).<br>REG_PWM2_DLOW < 0 x 00 00 02 21 |
| | 26 | invalid period duration set.<br>REG_PWM2_PD > 0 x FF FF FF FE |
| | 27 | invalid duty cycle (pulse width) set<br>REG_PWM2_DC > 0 x FF FF FF FE |
| | 28 to 30 | reserved |
| | 31 | A diagnostic message is pending.<br>The diagnostic byte 2 for PWM1 > 0 (→ see Diagnostic data of the module (page 3-9)). |

# 8 Error handling in the control interface/ check-back interface

## 8.1 Error messages of the module

In addition to the higher-level operation- and application relevant diagnostic messages (→ see page 3-9), each channel of the BL20-E-2CNT-2PWM also reports channel specific errors via the Process input/ check-back interface (page 3-14).

A distinction is made between:

- volatile status messages (**STS**)
  Display of current states (for example CNT/ PWM function enabled, CNT/ PWM active, CNT/PWM special function active, etc.), → see Process input/ check-back interface (page 3-14).

and

- non-volatile flags (**MSG**)
  Non-volatile storage of messages/ events (for example overflow, zero crossing, etc.) which could possibly get lost due to their time behavior.

### 8.1.1 Non-volatile flags (MSG)

**MSG for CNTx**

- CNT1: Process input/ check-back interface, byte 1
- CNT2: Process input/ check-back interface, byte 3

*Table 8-1: MSG for CNTx*

| Designation | Description | Triggering event |
|---|---|---|
| MSG_CNTx_CMP0 | Checking the count value for reaching the compare value 0. | REG_CNTx_CNT = REG_CNTx_CMP0 |
| MSG_CNTx_CMP1 | Checking the count value for reaching the compare value 1. | REG_CNTx_CNT = REG_CNTx_CMP1 |
| MSG_CNTx_UFLW | Checking the count value for an underflow. | REG_CNTx_CNT = REG_CNTx_UFLW |
| MSG_CNTx_OFLW | Checking the count value for an overflow. | REG_CNTx_CNT = REG_CNTx_OFLW |
| MSG_CNTx_ND | Checking the count value for a zero crossing. | REG_CNTx_CNT = 0 |
| MSG_CNTx_FQE | No counter pulse received within a defined time (definition in REG_CNTx_TO (page 9-6 or page 9-8)) although the count operation is enabled.<br>If count pulses were received, is only monitored while the count operation is enabled (STS_CNTx_RUN=1). | |
| MSG_CNTx_SFT | The event according to the parameterized special function occurred (→ see Special function of inputs Z1 and Z2 at CNTx (page 4-12)). | |
| MSG_CNTx_SW_LR | A software latch retrigger has been executed. | CNTx_SW_LR of the counter in the control interface changes from 0 → 1 |

### MSG for PWMx

- PWM1: Process input/ check-back interface, byte 4, bits 0 to 4
- PWM2: Process input/ check-back interface, byte 5, bits 0 to 4

*Table 8-2: MSG for PWMx*

| Designation | Description | Triggering event |
|---|---|---|
| MSG_PWMx_SW_LR | A software latch retrigger has been executed. | PWMx_SW_LR of the counter in the control interface changes from 0 → 1 |
| MSG_PWMx_NDDC | Checking the count value for a zero crossing. | MSG_PWMx_NDDC = 1 if REG_CNTx_CNTDC = 0 |
| MSG_PWMx_SFKT | The event according to the parameterized special function occurred (→ see Special function of inputs Z1 and Z2 for the PWM (page 5-11)). | |
| MSG_PWMx_DO_ERR | One of the outputs Px (page 3-9) or Dx (page 3-9) of the corresponding PWMx-channel sent an error. | Short-circuit at one of the outputs of the PWM channels. |

### 8.1.2 Storage of messages (MSG)

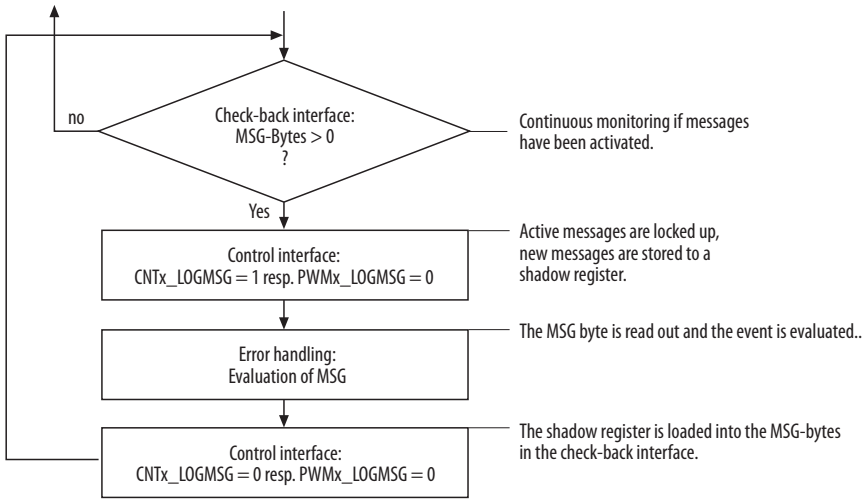
In order to guarantee that all events or state changes which effect the MSG bits can be registered at any time, the following behaviour was implemented:

- With the change from 0→ 1 in the control bits CNTx_LOGMSG or respectively PWMx_LOGMSG in the Process input/ check-back interface (→ see page 3-14), all states of the MSG bits are held.
- If events which would cause messages would occur in the meantime, they would get lost.
- To avoid this, a "shadow register" (REG_CNTx_LOGMSG, page 9-7 ff. or respectively REG_PWMx_LOGMSG, page 9-9 ff.) was created, in which, as long as the message bits are held, newly occurring messages (MSG) are stored.
- The basic status of the shadow register is: „all MSG = 0“.
- The states of the MSG bits, held in the process input data, can now be read out without newly incoming MSG going lost.

**Resetting the control bits**

- If now the control bits CNTx_LOGMSGor PWMx_LOGMSG in the Process output/ control interface (→ see page 3-19) are reset through 1 → 0, the messages that were meanwhile recorded in the shadow register are copied to the MSG bits in the Process input/ check-back interface (page 3-14).
- By means of this procedure, MSG bits of the Process input/ check-back interface (page 3-14) can be read out or respectively set back without loosing messages.

*Figure 8-1: Flow chart for the storage of MSG*



# 9 Register interface

## 9.1 Internal registers - reading and writing

This module provides a universal register interface that enables access to up to 128 32-bit-registers.

### 9.1.1 Write access

Write access is realized using REG_WR_DATA, Byte 0 to REG_WR_DATA, Byte 3 via the Process output/ control interface (page 3-19).

For write access, it must be ensured beforehand that the register write interface is in the default status and that there is no write access operation active. This is ensured if REG_WR=0 in the process output data, and is confirmed in the process input data with REG_WR_AKN=0.

Write access is now possible.

Therefore, the following values must be transferred with the process output data:

- the address of the register to be written in REG_WR_ADR, Process output/ control interface (page 3-19):
- the value to be written in REG_WR_DATA, Byte 0 to REG_WR_DATA, Byte 3, Process output/ control interface (page 3-19)
- the write command with REG_WR=1 Process output/ control interface (page 3-19)

The module acknowledges the processing of the write command by setting the acknowledge bit REG_WR_AKN=1 in the Process input/ check-back interface (page 3-14).

If register was successfully written, this is confirmed in the Process input/ check-back interface (page 3-14) with REG_WR_ACCEPT=1.

The write operation must then be terminated by REG_WR=0. This is done to resume the default state.

**Abort of the write command**

REG_WR_ACCEPT=0 indicates that the register could not be written (no access authorization, faulty value range,...).

**Example for a write access**

Writing the lower count limit „0“ of counter 1 in register no. 36 (0×24) REG_CNT1_LOLIMIT.

Write access:

**1** Address of the register to be written:
Process output/ control interface (page 3-19) →
REG_WR_ADR = 36 (0×24)

**2** value to be written:
lower count limit = 0
Process output/ control interface (page 3-19) →
REG_WR_DATA, Byte 0 = 00
REG_WR_DATA, Byte 1 = 00
REG_WR_DATA, Byte 2 = 00
REG_WR_DATA, Byte 3 = 00

**3** Process output/ control interface (page 3-19) →
REG_WR = 0 → 1
The write operation is enabled.

### 9.1.2 Read access

Read access to an optional register is done via the Process input/ check-back interface (page 3-14) as well as via the Process output/ control interface (page 3-19).

The following entries have to be carried out in the Process output/ control interface (page 3-19):

- Definition of the address of the register to be read out: REG_RD_ADR

The following entries are carried out by the module in the Process input/ check-back interface (page 3-14):

- As an acknowledge, REG_RD_ADR contains the address of the register to be read
- an error-free reading of the register is confirmed by REG_RD_ABORT = 0
- the read register content is shown in REG_RD_DATA, Byte 0 to REG_RD_DATA, Byte 3

**Abort of the read command**

REG_RD_ABORT = 1 shows that the register could not be read.

In case of a missed register access, REG_RD_ADR of the process input data contains the address that could not be accessed successfully.

The user data is then set to ZERO.

**Example for a read access**

Reading the actual count value of counter 1 from register no. 32 (0×20) REG_CNT1_CNT.

Read access:

1. Address of the register to be read:
   Process output/ control interface (page 3-19) →
   REG_RD_ADR = 32 (0×20)
2. Feedback:
   Process input/ check-back interface (page 3-14) →
   REG_RD_ADR = 32 (0×20)
3. Feedback:
   Process input/ check-back interface (page 3-14) →
   REG_RD_ABORT = 0
   The read access was successful.
4. read value:
   Process input/ check-back interface (page 3-14) →
   Example:
   REG_RD_DATA, Byte 0 = 27
   REG_RD_DATA, Byte 1 = 10
   REG_RD_DATA, Byte 2 = 00
   REG_RD_DATA, Byte 3 = 00

## 9.2 Register description and register access

### 9.2.1 Register interface

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| **Standard registers** | | | | | | | | | |
| - | 0×00 | | | | | | | | |
| REG_MAGIC_NO | 0×01 | Magic number (internal use) | | 0×aa55cc33 | | | RD | | |
| REG_HW_VER | 0×02 | hardware-version | | | | | RD | | |
| REG_SW_VER | 0×03 | Firmware version | | | | | RD | | |
| REG_SF | 0×04 | Special function register | | | V | | WR | | |
| REG_IF_VER | 0×05 | Version of the register interface | | | | | RD | | |
| | ... | reserved | | | | | RD | | |
| REG_CONFIG_ERRSTS | 0×0A | report of configuration errors | | | V | | RD | | |
| | ... | reserved | | | | | | | |
| REG_DATA_IN1, byte 3-0 | 0×0C | Process input 1 | 32 bit unsigned | | V | | RD | | |
| REG_DATA_IN2, byte 7-4 | 0×0D | ... | | | V | | RD | | |
| REG_DATA_IN3, byte 11-8 | 0×0E | | | | V | | RD | | |
| REG_DATA_IN4, byte 15-12 | 0×0F | | | | V | | RD | | |
| REG_DATA_IN5, byte 19-16 | 0×10 | | | | V | | RD | | |
| REG_DATA_IN6, byte 23-20 | 0×11 | Process input 6 | | | V | RD | | | |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_DATA_OUT1, byte 3-0 | 0×12 | Process output 1 | | | V | RD | | | |
| REG_DATA_OUT2, byte 7-4 | 0×13 | ... | | | V | RD | | | |
| REG_DATA_OUT3, byte 11-8 | 0×14 | | | | V | RD | | | |
| REG_DATA_OUT4, byte 15-12 | 0×15 | | | | V | RD | | | |
| REG_DATA_OUT5, byte 19-16 | 0×16 | | | | V | RD | | | |
| REG_DATA_OUT6, byte 23-20 | 0×17 | Process output 6 | | | V | RD | | | |
| REG_DIAG1, byte 3-0 | 0×18 | Diagnostic data 1 | | | V | RD | | | RD |
| | 0×19 to 1B | reserved | | | V | | | | |
| REG_PARA1, byte 3-0 | 0×1C | parameter data 1 | | (0×00 00 00 00) | NV | WR | RD | WR | |
| REG_PARA2, byte 7-4 | 0×1D | ... | 32 bit unsigned | (0×00 00 00 00) | NV | WR | RD | WR | |
| REG_PARA3, byte 11-8 | 0×1E | | | (0×00 00 00 00) | NV | WR | RD | WR | |
| REG_PARA4, byte 15-12 | 0×1F | parameter data 3 | | (0×00 00 00 00) | NV | WR | RD | WR | |
| **Register CNT1** | | | | | | | | | |
| REG_CNT1_CNT | 0×20 | Actual binary value of the CNT1 | | | V | WR | RD | | |
| REG_CNT1_MV | 0×21 | measured value CNT1 | 32 bit unsigned | | V | | RD | | |
| ... | 0×22 | reserved | - | - | - | - | - | - | - |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_CNT1_LOADVAL | 0×23 | load value CNT1 | 32 bit signed | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT1_LOLIMIT | 0×24 | Lower count limit CNT1 | | (0×80 00 00 00) | NV | WR | RD | | |
| REG_CNT1_HILIMIT | 0×25 | Upper count limit CNT1 | | 0×7F FF FF FF | NV | WR | RD | | |
| REG_CNT1_CMP0 | 0×26 | Compare value 0 CNT1 | | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT1_CMP1 | 0×27 | Compare value 1 CNT1 | | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT1_LATCH | 0×28 | Buffer | 32 bit signed | - | V | | RD | | |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_CNT1_INTTIME | 0×29 | Integration time CNT1 in 10 ms/ bit | 32 bit unsigned | 0×00 00 00 0A (100 ms) | NV | WR | RD | | |
| REG_CNT1_MUL | 0×2A | Factor CNT1 | | (0×00 00 00 01) | NV | WR | RD | | |
| REG_CNT1_DIV | 0×2B | Divisor CNT1 | | (0×00 00 00 01) | NV | WR | RD | | |
| REG_CNT1_IPI | 0×2C | Pulses per integration time | | - | V | | RD | | |
| REG_CNT1_TO | 0×2D | Time out CNT1 in 10 ms/ bit | | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT1_LOGMSG | 0×2E | Buffer of the MSG register at MSGLOG (→ see CNTx_LOGMSG (page 3-20)) | | - | V | | RD | | |
| REG_CNT1_DO1_HYS | 0×2F | Hysteresis of the D1 and STS_DBP1 for CNT1 | | 0×00 00 00 0A (10 pulses) | NV | WR | RD | | |
| REG_CNT1_DO1_IMP | 0×30 | Pulse time for a pulse output at D1 in 10 ms/ bit | | 0×00 00 00 0A (100 ms) | NV | WR | RD | | |
| | 0×31 to 0×3F | reserved | | - | - | | | | |
| **Register CNT2** | | | | | | | | | |
| REG_CNT2_CNT | 0×40 | Actual binary value of the CNT2 | | - | V | WR | RD | | |
| REG_CNT2_MV | 0×41 | Measured value CNT2 | 32 bit unsigned | | V | | RD | | |
| ... | 0×42 | reserved | | | - | - | - | | |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_CNT2_LOADVAL | 0×43 | Load value CNT2 | 32 bit signed | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT2_LOLIMIT | 0×44 | lower count limit CNT2 | | (0×080 00 00 00) | NV | WR | RD | | |
| REG_CNT2_HILIMIT | 0×45 | Upper count limit CNT2 | | 0×7F FF FF FF | NV | WR | RD | | |
| REG_CNT2_CMP0 | 0×46 | Compare value 0 CNT2 | | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT2_CMP1 | 0×47 | Compare value 1 CNT2 | | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT2_LATCH | 0×48 | Buffer | | - | V | | RD | | |
| REG_CNT2_INTTIME | 0×49 | Integration time CNT2 in 10 ms/bit | 32 bit unsigned | 0×00 00 00 0A (100 ms) | NV | WR | RD | | |
| REG_CNT2_MUL | 0×4A | Factor CNT2 | | (0×00 00 00 01) | NV | WR | RD | | |
| REG_CNT2_DIV | 0×4B | Divisor CNT2 | | (0×00 00 00 01) | NV | WR | RD | | |
| REG_CNT2_IPI | 0×4C | Pulses per integration time | | - | V | | RD | | |
| REG_CNT2_TO | 0×4D | Time out CNT2 in 10 ms/ bit | | (0×00 00 00 00) | NV | WR | RD | | |
| REG_CNT2_LOGMSG | 0×4E | Buffer of the MSG register at MSGLOG | | - | V | | RD | | |
| REG_CNT2_DO1_HYS | 0×4F | Hysteresis of the D2 and STS_DBP2 for CNT2 | | 0×00 00 00 0A (10 pulses) | NV | WR | RD | | |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_CNT2_DO2_IMP | 0×40 | Pulse time for a pulse output at D2 in 10 ms/ bit | 32 bit unsigned | 0×00 00 00 0A (100 ms) | NV | WR | RD | | |
| | 0×51 to 0× 5F | reserved | | | | | | | |
| **Register PWM1** | | | | | | | | | |
| REG_PWM1_PD | 0×60 | Period duration PWM1 in 41,667 ns/ bit | 32 bit unsigned | 0×00 00 5D C0 (1000 Hz) | V | WR | RD | | |
| REG_PWM1_DC | 0×61 | Mark-to-space ratio PWM1, in 23,28 × $10^{-9}$ %/ bit | | 0×7F FF FF FF (50 %) | V | WR | RD | | |
| REG_PWM1_DHIGH | 0×62 | Pulse duration PWM1 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | V | WR | RD | | |
| REG_PWM1_DLOW | 0×63 | Period duration PWM1 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | V | WR | RD | | |
| REG_PWM1_CNTSV | 0×64 | Load value of the pulses to be given out | | 0×00 00 27 10 (10000 pulses) | NV | WR | RD | | |
| REG_PWM1_CNTDC | 0×65 | Number of pulse which still have to be given out PWM1 | | - | V | | RD | | |
| REG_PWM1_LATCH | 0×66 | Buffer PWM1 | | - | V | | RD | | |
| REG_PWM1_LOGMSG | 0×67 | Buffer of the MSG bits at MSGLOG, PWM1 | | - | V | | RD | | |
| REG_PWM1_PD_RV | 0×68 | Start value after reset: Period duration PWM1 in 41,667 ns/ bit | | 0×00 00 5D C0 (1000 Hz) | NV | WR | RD | | |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_PWM1_DC_RV | 0×69 | Start value after reset: mark-to-space ratio PWM1 in 23,28 × $10^{-9}$ %/ bit | 32 bit unsigned | 0×7F FF FF FF (50 %) | NV | WR | RD | | |
| REG_PWM1_DHIGH_RV | 0×6A | Start value after reset: Pulse duration PWM1 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | NV | WR | RD | | |
| REG_PWM1_DLOW_RV | 0×6B | Start value after reset: Space duration PWM1 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | NV | WR | RD | | |
| | 0×6C to 0×6F | reserved | | | | | | | |
| **Register PWM2** | | | | | | | | | |
| REG_PWM2_PD | 0×70 | Period duration PWM2 in 41,667 ns/ bit | 32 bit unsigned | 0×00 00 5D C0 (1000 Hz) | V | WR | RD | | |
| REG_PWM2_DC | 0×71 | Mark-to-space ratio PWM2, in 23,28 × $10^{-9}$ %/ bit | | 0×7F FF FF FF (50 %) | V | WR | RD | | |
| REG_PWM2_DHIGH | 0×72 | Pulse duration PWM2 in 41,667ns/ bit | | 0×00 00 2E E0 (500 µs) | V | WR | RD | | |
| REG_PWM2_DLOW | 0×73 | Space duration PWM2 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | V | WR | RD | | |
| REG_PWM2_CNTSV | 0×74 | Load value of the pulses to be given out | | 0×00 00 27 10 (10000 pulses) | NV | WR | RD | | |
| REG_PWM2_CNTDC | 0×75 | Number of pulse which still have to be given out PWM1 | | - | V | | RD | | |
| REG_PWM2_LATCH | 0×76 | Buffer PWM2 | | - | V | | RD | | |

| Designation | No. | Description | Format | Default (HEX) | Storage in module | Process output | Process input | Parameters | diagnostic messages |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | V = volatile<br>NV = non-volatile<br>RD = read access<br>WR = write access | | | | |
| REG_PWM2_LOGMSG | 0×77 | Buffer of the MSG register at MSGLOG, PWM2 | 32 bit unsigned | - | V | | RD | | |
| REG_PWM2_PD_RV | 0×78 | Start value after reset: Period duration PWM2 in 41,667 ns/ bit | | 0×00 00 5D C0 (1000 Hz) | NV | WR | RD | | |
| REG_PWM2_DC_RV | 0×79 | Start value after reset: mark-to-space ratio PWM2 in 23,28 × $10^{-9}$ %/ bit | | 0×7F FF FF FF (50 %) | NV | WR | RD | | |
| REG_PWM2_DHIGH_RV | 0×7A | Start value after reset: Pulse duration PWM2 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | NV | WR | RD | | |
| REG_PWM2_DLOW_RV | 0×7B | Start value after reset: Space duration PWM2 in 41,667 ns/ bit | | 0×00 00 2E E0 (500 µs) | NV | WR | RD | | |
| | 0×7C to 0×7F | reserved | | | | | | | |

**Note**

Non-volatile registers can be written for a maximum of 100.000 times.

### Special function register/ resetting the register interface

If the Special Function Register register no. 0×4 REG_SF is written with

LD20 = 0×4C443230

or

ld20 = 0×6C643230

the default values of all non-volatile registers (→ see Register interface (page 9-4)) **and**of the parameters (→ see Parameter data of the module (page 3-10)) are reset and the volatile registers REG_PWMx_PD, REG_PWMx_DC, REG_PWMx_DHIGH and REG_PWMx_DLOW (→ see page 9-10 ff.) are loaded with the content of the respective reset value registers (for example REG_PWM1_PD_RV).

Values that have been entered before get lost.

If the Special Function Register register no. 0×4 REG_SF is written with

LD48 = 0x4C443230

or

ld48 = 0×6C643438

the default values of all non-volatile registers (→ see Register interface (page 9-4)), **not** those of the parameters, are reset and the volatile registers REG_PWMx_PD, REG_PWMx_DC, REG_PWMx_DHIGH and REG_PWMx_DLOW (→ see page 9-10 ff.) are loaded with the content of the respective reset value registers (for example REG_PWM1_PD_RV).

**Note**

Values that have been entered before get lost.

# 10 Representation of the BL20-E-2CNT-2PWM in PROFIBUS-DPV1

## 10.1 Process data of the module in PROFIBUS-DPV1

**Hinweis**

The structure of the process data bits in PROFIBUS-DPV1 basically corresponds to the general structure of the process data (→ see chapter 3, section Process data of the module (page 3-14)).

It has thus to be observed that the structure of the user data words in PROFIBUS-DP differs from the general structure.

### 10.1.1 Process input/ check-back interface

**Note**

Please find the meaning of the bits in the general description of the modules Process input/ check-back interface in chapter 3 from page 3-14.

| | | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Status messages | CNTx (page 3-15) | 0 | A1 | B1 | Z1 | STS_CNT1_DIR | STS_CNT1_LOGMSG | STS_CNT1_SFKT_EN | STS_CNT1_RUN | STS_CNT1_GENERAL_EN |
| | | 1 | MSG_CNT1_SW_LR | MSG_CNT1_SFKT | MSG_CNT1_FQE | MSG_CNT1_ND | MSG_CNT1_OFLW | MSG_CNT1_UFLW | MSG_CNT1_CMP1 | MSG_CNT1_CMP0 |
| | | 2 | A2 | B2 | Z2 | STS_CNT2_DIR | STS_CNT2_LOGMSG | STS_CNT2_SFKT_EN | STS_CNT2_RUN | STS_CNT2_GENERAL_EN |
| | | 3 | MSG_CNT2_SW_LR | MSG_CNT2_SFKT | MSG_CNT2_FQE | MSG_CNT2_ND | MSG_CNT2_OFLW | MSG_CNT2_UFLW | MSG_CNT2_CMP1 | MSG_CNT2_CMP0 |
| | PWMx (page 3-16) | 4 | STS_PWM_LOGMSG | STS_PWM_SFKT_EN | STS_PWM1_RUN | STS_PWM1_GENERAL_EN | MSG_PWM1_DO_ERR | MSG_PWM1_SFKT | MSG_PWM1_NDDC | MSG_PWM1_SW_LR |
| | | 5 | STS_PWM2_LOGMSG | STS_PWM2_SFKT_EN | STS_PWM2_RUN | STS_PWM2_GENERAL_EN | MSG_PWM2_DO_ERR | MSG_PWM2_SFKT | MSG_PWM2_NDDC | MSG_PWM2_SW_LR |
| register access (page 9-2) and DOs | | 6 | REG_WR_ACCEPT | REG_WR_AKN | REG_RD_ABORT | STS_CONFIG_ERR | STS_DBP2 | D2 | STS_DBP1 | D1 |
| | | 7 | X | REG_RD_ADR | | | | | | |
| User data (page 3-18) | | 8 | REG_RD_DATA, Byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 11 | REG_RD_DATA, Byte 0 | | | | | | | |
| | | 12 | AUX_REG1_RD_DATA, Byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 15 | AUX_REG1_RD_DATA, Byte 0 | | | | | | | |
| | | 16 | AUX_REG2_RD_DATA, Byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 19 | AUX_REG2_RD_DATA, Byte 0 | | | | | | | |
| | | 20 | AUX_REG3_RD_DATA, Byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 23 | AUX_REG3_RD_DATA, Byte 0 | | | | | | | |

### 10.1.2 Process output/ control interface

**Hinweis**

Please find the meaning of the bits in the general description of the modules Process output/ control interface in chapter 3 from page 3-19.

| | | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Control bytes | CNT x (page 3-19) | 0 | X | CNT1 _ SINGLE | CNT1_ SW_LR | CNT1_SFKT _DISABLE | X | CNT1_ LOGMSG | CNT1_ ENABLE | CNT1_ GENERAL_ DISABLE |
| | | 1 | X | CNT2 _ SINGLE | CNT2_ SW_LR | CNT2_SFKT _DISABLE | X | CNT2_ LOGMSG | CNT2_ ENABLE | CNT2_ GENERAL_ DISABLE |
| | PWM x (page 10-4) | 2 | X | PWM1_ SINGLE | PWM1_ SW_LR | PWM1_ SFKT_ DISABLE | X | PWM1_ LOGMSG | PWM1_ ENABLE | PWM1_ GENERAL_ DISABLE |
| | | 3 | X | PWM2_ SINGLE | PWM2_ SW_LR | PWM2_ SFKT_ DISABLE | X | PWM2_ LOGMSG | PWM2_ ENABLE | PWM2_ GENERAL_ DISABLE |
| | DOs | 4 | X | X | SET_P2 | SET_ D2 | X | X | SET_P1 | SET_ D1 |
| register access (page 9-2) | | 5 | REG_WR | X | X | X | X | AUX_REG3_ WR_EN | AUX_REG2_ WR_EN | AUX_REG1_ WR_EN |
| | | 6 | X | REG_WR_ADR | | | | | | |
| | | 7 | X | REG_RD_ADR | | | | | | |
| User data | | 8 | REG_WR_DATA, Byte 3 | | | | | | | |
| | | 9 | REG_WR_DATA, Byte 2 | | | | | | | |
| | | 10 | REG_WR_DATA, Byte 1 | | | | | | | |
| | | 11 | REG_WR_DATA, Byte 0 | | | | | | | |
| | | 12 | AUX_REG1_WR_DATA, byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 15 | AUX_REG1_WR_DATA, byte 0 | | | | | | | |
| | | 16 | AUX_REG2_WR_DATA, byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 19 | AUX_REG2_WR_DATA, byte 0 | | | | | | | |
| | | 20 | AUX_REG3_WR_DATA, byte 3 | | | | | | | |
| | | ... | ... | | | | | | | |
| | | 23 | AUX_REG3_WR_DATA, byte 0 | | | | | | | |

X = reserved

## 10.2 Diagnostics of the module in DPV1

**Note**

The module BL20-E-2CNT-2PWM can only be used with the BL20 DPV1 gateways. These gateways support the diagnostic function according to PROFIBUS DP specification IEC61158, type 3.
Please read the corresponding gateway manuals for the description of the diagnostic message structure of the DPV1 gateways (→ see Additional documentation (page 1-2)).

### 10.2.1 DPV1 error codes

*Table 10-1: DPV1 error codes*

| Error code (acc. to DPV1 standard) | diagnostic messages | Channel | Message |
|---|---|---|---|
| 4 | overload | CH2 | P1_DIAG |
| | | CH4 | P2_DIAG |
| | | CH3 | D1_DIAG |
| | | CH5 | D2_DIAG |
| 16 | parameterization error | CH0 | CNT1_PAR _ERR |
| | | CH1 | CNT2_PAR _ERR |
| | | CH2 | PWM1_PAR _ERR |
| | | CH4 | PWM2_PAR _ERR |
| 27 | unknown error | reserved | HW_ERR |

### 10.2.2 Diagnostic data

The module's diagnostic data contain error messages that are operation and application relevant for the control system. 4 bytes are used to transfer the diagnostic data.

**Hinweis**

A more detailed description of the module's diagnostic data can be found in the field bus independent module description (chapter 3, section Diagnostic data of the module (page 3-9)).

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | HW_ERR | CNT1_ PAR_ERR | X | X | X | X | X | X |
| 1 | HW_ERR | CNT2_ PAR_ERR | X | X | X | X | X | X |
| 2 | HW_ERR | PWM1_ PAR_ERR | X | X | X | X | P1_DIAG | D1_DIAG |
| 3 | HW_ERR | PWM2_ PAR_ERR | X | X | X | X | P2_DIAG | D2_DIAG |

## 10.3 Parameter in DPV1

**Note**

The module BL20-E-2CNT-2PWM can only be used with the BL20 DPV1 gateways. These gateways support the parameterization according to PROFIBUS DP specification IEC61158, type 3.
Please read the corresponding gateway manuals for the description of the diagnostic message structure of the DPV1 gateways ((→ siehe Additional documentation (page 1-2)).

### 10.3.1 Parameter data of the module

**Note**

A more detailed description of the module's parameter data can be found in the field bus independent module description (chapter 3, section Parameter data of the module (page 3-10)).

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | input A1 | input B1 | input Z1 | X | diagnostic CNT1 | measure-ment mode CNT1 | main count direction CNT1 | |
| 1 | filter Z1 | | filter A1, B1 | | X | pull up Z1 | X | threshold input A,B,Z CNT1 |
| 2 | mode Z1 | | | | mode CNT1 | | | |
| 3 | input A2 | input B2 | input Z2 | X | diagnostic | measure-ment mode CNT2 | main count direction CNT2 | |
| 4 | filter Z2 | | filter A2, B2 | | X | pull up Z2 | X | threshold input A,B,Z CNT2 |
| 5 | mode Z2 | | | | mode CNT2 | | | |
| 6 | diagnostic PWM1 | X | mode D1 | | | | | |
| 7 | DBP1 STS MODE | | substitute value P1 | substitute value D1 | mode PWM1 | | | |
| 8 | diagnostic PWM2 | X | mode D2 | | | | | |
| 9 | DBP2 STS MODE | | substitute value P2 | substitute value D2 | mode PWM2 | | | |
| 10 | X | ADR AUX REG1 RD DATA | | | | | | |
| 11 | X | ADR AUX REG2 RD DATA | | | | | | |
| 12 | X | ADR AUX REG3 RD DATA | | | | | | |
| 13 | X | ADR AUX REG1 WR DATA | | | | | | |
| 14 | X | ADR AUX REG2 WR DATA | | | | | | |
| 15 | X | ADR AUX REG3 WR DATA | | | | | | |

X = reserved

# 11 Representation of the BL20-E-2CNT-2PWM in CANopen

## 11.1 Objects for counter modules

### 11.1.1 General object overview for counter modules

*Table 11-1: General object overview for counter modules*

| | | Object | Name | page |
|---|---|---|---|---|
| Manufacturer specific objects | CNTx | $5800_{hex}$ | not supported | |
| | | $5801_{hex}$ | Encoder Config | page 11-4 |
| | | $5802_{hex}$ | Encoder Status | page 11-6 |
| | | $5803_{hex}$ | Encoder Flags | page 11-7 |
| | | $5804_{hex}$ | Encoder Diag | page 11-8 |
| | | $5805_{hex}$ | Encoder Native Status | page 11-9 |
| | | $5806_{hex}$ | Encoder Optional Status | page 11-10 |
| | | $5808_{hex}$ | Encoder Control | page 11-11 |
| | | $5810_{hex}$ | Encoder Load Prepare Value | page 11-12 |
| | | $5811_{hex}$ | Encoder Pulse Width | page 11-13 |
| | | $5820_{hex}$ | Measuring Integration Time | page 11-14 |
| | | $5821_{hex}$ | not supported | |
| | | $5822_{hex}$ | | |
| | | $5823_{hex}$ | | |
| | | $5824_{hex}$ | Encoder Measuring Divisor | page 11-15 |
| | | $5825_{hex}$ | Encoder Measuring Factor | page 11-16 |
| | | $5827_{hex}$ | Encoder Measuring Time Out | page 11-17 |
| | | $5830_{hex}$ | Encoder Measuring Value | page 11-18 |
| | | $5831_{hex}$ | Encoder Latch Value | page 11-19 |

*Table 11-1: General object overview for counter modules*

| | | Object | Name | page |
|---|---|---|---|---|
| Manufacturer specific objects | PWMx | $5901_{hex}$ | PWM Config | page 11-20 |
| | | $5902_{hex}$ | PWM Status | page 11-22 |
| | | $5903_{hex}$ | PWM Flags | page 11-23 |
| | | $5904_{hex}$ | PWM Diag | page 11-24 |
| | | $5908_{hex}$ | PWM Control | page 11-25 |
| | | $5910_{hex}$ | PWM Load Prepare Value | page 11-26 |
| | | $5913_{hex}$ | PWM Duty Cycle | page 11-27 |
| | | $5920_{hex}$ | PWM Period Duration | page 11-28 |
| | | $5931_{hex}$ | PWM Latch Value | page 11-29 |
| Objects according to DS 406 + offset (0×800) | | $6810_{hex}$ | Preset Value For Multi-Sensor Devices | page 11-30 |
| | | $6820_{hex}$ | Preset Value For Multi-Sensor Devices | page 11-31 |
| | | $6B00_{hex}$ | CAM 1 State Register | page 11-32 |
| | | $6B00_{hex}$ | CAM 1 Enable Register | page 11-33 |
| | | $6B00_{hex}$ | Cam 1 Polarity Register | page 11-34 |
| | | $6B10_{hex}$ | Cam 1 Low Limit | page 11-35 |
| | | $6B20_{hex}$ | Cam 1 High Limit | page 11-36 |
| | | $6B30_{hex}$ | Cam 1 Hysteresis | page 11-37 |
| | | $6C00_{hex}$ | Area State Register | page 11-38 |
| | | $6C01_{hex}$ | Work Area Low Limit | page 11-39 |
| | | $6C02_{hex}$ | Work Area High Limit | page 11-40 |
| | | Diagnostics | | |
| | | $6FFF_{hex}$ | Device Type | page 11-41 |

### 11.1.2 Object descriptions

**Object 5801hex – Encoder Config**

The object $5801_{hex}$ affects the configuration parameters of CNTx:

- output parameters
- sensor and input filter
- sensor parameters
- behavior on failure of the higher-level PLC

Write accesses initiate a parameter update via the internal module bus of the BL20 station. The parameter is stored as a non-volatile parameter in the BL20 gateway and is restored with every node reset.

*Table 11-2: Object $5801_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Config |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | input A1 (0×00) | input B1 (0×00) | input Z1 (0×00) | X | diagnostic CNT1 (0×00) | measure-ment mode CNT1 (0×00) | main count direction CNT1 (0×00) | |
| | 1 | **Bit 15** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | filter Z1 (0×00) | | filter A1, B1 (0×00) | | X | pull up Z1 (0×00) | X | threshold input A,B,Z CNT1 (0×00) |
| | 2 | **Bit 23** | **Bit 22** | **Bit 21** | **Bit 20** | **Bit 19** | **Bit 18** | **Bit 17** | **Bit 16** |
| | | mode Z1 (0×03) | | | | mode CNT1 (0×00) | | | |
| | 3 | **Bit 31 (msb)** | **Bit 30** | **Bit 29** | **Bit 28** | **Bit 27** | **Bit 26** | **Bit 25** | **Bit 24** |
| | | X | ADR AUX REG1 RD DATA (0×20) | | | | | | |
| **n + 1** = CNT2 | 0 | input A2 (0×00) | input B2 0×00) | input Z2 0×00) | X | diagnostic CNT2 (0×00) | measure-ment mode CNT2 (0×00) | main count direction CNT2 (0×00) | |
| | 1 | **Bit 15** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | filter Z2 (0×00) | | filter A2, B2 (0×00) | | X | pull up Z2 (0×00) | X | threshold input A,B,Z CNT2 (0×00) |
| | 2 | **Bit 23** | **Bit 22** | **Bit 21** | **Bit 20** | **Bit 19** | **Bit 18** | **Bit 17** | **Bit 16** |
| | | mode Z2 (0×03) | | | | mode CNT2 (0×00) | | | |
| | 3 | **Bit 31 (msb)** | **Bit 30** | **Bit 29** | **Bit 28** | **Bit 27** | **Bit 26** | **Bit 25** | **Bit 24** |
| | | X | ADR AUX REG3 RD DATA (0×40) | | | | | | |

( ) = default parameterization

**Note**

A more detailed description of the module's parameter data can be found in the field bus independent module description (chapter 3, section Parameter data of the module (page 3-10)).

### Object 5802hex – Encoder Status

Status displays of the CNTx from the process input data.

The object $5802_{hex}$ supplies the following status messages:

- count direction
- status of in- and outputs
- operation status of the counter

*Table 11-3: Object $5802_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Status |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

The meaning of the status data bytes generally corresponds to the general description of the module (→ see Process input/ check-back interface (page 3-14)).

**exception**:

The count direction is not reported via a common bit (STS_CNTx_DIR, page 3-14) but using the two bits STS_CNT1_UP and

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 (msb) | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | STS_CNT1_DN | STS_CNT1_UP | STS_CNT1_SFKT_EN | STS_DBP1 | D1 | STS_CNT1_GENERAL_EN | Z1 | STS_CNT1_RUN |
| **n + 1** = CNT2 | 0 | STS_CNT2_DN | STS_CNT2_UP | STS_CNT2_SFKT_EN | STS_DBP2 | D2 | STS_CNT2_GENERAL_EN | Z2 | STS_CNT2_RUN |

**Note**

A more detailed description of the module's status messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 5803hex – Encoder Flags

Das Object $5803_{hex}$ supplies the following messages via the process input data:

- reaching of limit values
- execution of a SW latch retrigger
- an event defined as special function (SFKT) has occurred
- a time out in the count pulse measurement occurred

*Table 11-4: Object $5803_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Flags |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Access PDO | 0×01 to 0×47 | r |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

The meaning of the data bytes in the process input/ check-back interface generally corresponds to the general description of the module (→ see Process input/ check-back interface (page 3-14)).

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 (msb) | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | MSG_ CNT1_ND | MSG_ CNT1_ UFLW | MSG_ CNT1_ OFLW | MSG_ CNT1_ CMP1 | MSG_ CNT1_ CMP0 | MSG_ CNT1_SW_ LR | MSG_ CNT1_ SFKT | MSG_ CNT1_ FQE |
| **n + 1** = CNT2 | 0 | MSG_ CNT2_ND | MSG_ CNT2_ UFLW | MSG_ CNT2_ OFLW | MSG_ CNT2_ CMP1 | MSG_ CNT2_ CMP0 | MSG_ CNT2_SW_ LR | MSG_ CNT2_ SFKT | MSG_ CNT2_ FQE |

**Note**

A more detailed description of the module's messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 5804hex – Encoder Diag

The object $5804_{hex}$ reads error messages of CNTx from the module's diagnostic data.

*Table 11-5: Object $5804_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Diag |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

The meaning of the status data bytes generally corresponds to the general description of the module (→ see Diagnostic data of the module).

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 (msb) | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | HW_ERR | CNT1_ PAR_ERR | CNT1_ERR_ UFLW | CNT1_ERR_ OFLW | reserved | | | |
| **n + 1** = CNT2 | 0 | HW_ERR | CNT2_ PAR_ERR | CNT2_ERR_ UFLW | CNT2_ERR_ OFLW | reserved | | | |

**Note**

A more detailed description of the module's diagnostic data can be found in the field bus independent module description (chapter 3, section Diagnostic data of the module (page 3-9)).

### Object 5805hex – Encoder Native Status

The object $5805_{hex}$ reads the counter's status byte and the flag byte (B1, B0) from the Process input/ check-back interface of the module. The following operation states are reported.

*Table 11-6: Object $5805_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Native Status |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 16 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Structure of the data bytes**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | A1 | B1 | Z1 | STS_CNT1_DIR | STS_CNT1_LOGMSG | STS_CNT1_SFKT_EN | STS_CNT1_RUN | STS_CNT1_GENERAL_EN |
| | 1 | **Bit 15 (msb)** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | MSG_CNT1_SW_LR | MSG_CNT1_SFKT | MSG_CNT1_FQE | MSG_CNT1_ND | MSG_CNT1_OFLW | MSG_CNT1_UFLW | MSG_CNT1_CMP1 | MSG_CNT1_CMP0 |
| **n + 1** = CNT2 | **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0 (lsb)** |
| | 0 | A2 | B12 | Z2 | STS_CNT2_DIR | STS_CNT2_LOGMSG | STS_CNT2_SFKT_EN | STS_CNT2_RUN | STS_CNT2_GENERAL_EN |
| | 1 | **Bit 15 (msb)** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | MSG_CNT2_SW_LR | MSG_CNT2_SFKT | MSG_CNT2_FQE | MSG_CNT2_ND | MSG_CNT2_OFLW | MSG_CNT2_UFLW | MSG_CNT2_CMP1 | MSG_CNT2_CMP0 |

**Note**

A more detailed description of the module's messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 5806hex – Encoder Optional Status

The object $5806_{hex}$ supplies the following status messages:

*Table 11-7: Object $5806_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Optional Status |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | reserved | | | | | | STS_DBP1 | D1 |
| **n + 1** = CNT2 | 0 | reserved | | | | | | STS_DBP2 | D2 |

**Note**

A more detailed description of the module's messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 5808hex – Encoder Control

The object $5808_{hex}$ provides the following control functions for CNTx (Process output/ control interface (page 3-19)):

- general enabling or disabling of the count function
- start/ stop of the count operation
- storing of Error messages of the module (page 8-2) for readout without data loss
- enabling or disabling of the special function
- executing a software latch retrigger
- setting the single or continuous count

*Table 11-8: Object $5808_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Control |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Access PDO | 0×01 to 0×47 | r |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | reserved | CNT1_ SINGLE | CNT1_ SW_LR | CNT1_ SFKT_ DISABLE | reserved | CNT1_ LOGMSG | CNT1_ ENABLE | CNT1_ GENERAL_ DISABLE |
| **n + 1** = CNT2 | 0 | reserved | CNT2_ SINGLE | CNT2_ SW_LR | CNT2_ SFKT_ DISABLE | reserved | CNT2_ LOGMSG | CNT2_ ENABLE | CNT2_ GENERAL_ DISABLE |

**Note**

A more detailed description of the module's control bits can be found in the field bus independent module description (chapter 3, section Process output/ control interface (page 3-19)).

### Object 5810hex – Encoder Load Prepare Value

The object $5810_{hex}$ contains the load value (load value register, CNT1 no. 0×23, CNT2 no. 0×43) for the "prepared loading" of the counters. Setting the counter's count value to this value is event-driven.

*Table 11-9: Object $5810_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Load Prepare Value |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Integer 32 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | load value register of CNT1 (Register interface (page 9-4)) | REG_CNT1_LOADVAL (page 9-6) | 0×23 |
| **n + 1** = CNT2 | 0 to 3 | load value register of CNT2 (Register interface (page 9-4)) | REG_CNT2_LOADVAL (page 9-8) | 0×43 |

### Object 5811hex – Encoder Pulse Width

The object $5811_{hex}$ is used for setting the pulse duration. The time is set in 1 ms/ bit.

Value range: 0 ms to 65535 ms (1 min 5 s).

In the module, the time is stored with a time base of 10 ms/ bit. The gateway thus converts the value respectively.

*Table 11-10: Object $5811_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Pulse Width |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 16 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | register containing the pulse time for a pulse output at D1<br>(Register interface (page 9-4)) | REG_CNT1_DO1_IMP (page 9-7) | 0×30 |
| **n + 1** = CNT2 | 0 to 3 | register containing the pulse time for a pulse output at D2<br>(Register interface (page 9-4)) | REG_CNT2_DO2_IMP (page 9-9) | 0×50 |

### Object 5820hex – Measuring Integration Time

The object $5820_{hex}$ is used to set the integration time for the counters.

For the Frequency measurement (page 4-19), Revolutions speed measurement (page 4-22) the integration time is entered in 10 ms/ bit.

*Table 11-11: Object $5820_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Measuring Integration Time |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | Integration time CNT1 10 ms/ bit (Register interface (page 9-4)) | REG_CNT1_INTTIME (page 9-7) | 0×29 |
| **n + 1** = CNT2 | 0 to 3 | integration time CNT2 10 ms/ bit (Register interface (page 9-4)) | REG_CNT2_INTTIME (page 9-8) | 0×49 |

### Object 5824hex – Encoder Measuring Divisor

The object $5824_{hex}$ is used for scaling the measured value (→ see also Additional function: Measurement mode (page 4-18)).

*Table 11-12: Object $5824_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Measuring Divisor |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Structure of the data bytes**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | Divisor CNT1<br>(Register interface (page 9-4)) | REG_CNT1_DIV (page 9-7) | 0×2B |
| **n + 1** = CNT2 | 0 to 3 | Divisor CNT2<br>(Register interface (page 9-4)) | REG_CNT2_DIV (page 9-8) | 0×4B |

### Object 5825hex – Encoder Measuring Factor

The object $5825_{hex}$ is used for scaling the measured value (→ see also Additional function: Measurement mode (page 4-18)).

*Table 11-13: Object $5825_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Measuring Factor |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | Factor CNT1<br>(Register interface (page 9-4)) | REG_CNT1_MUL (page 9-7) | 0×2A |
| **n + 1** = CNT2 | 0 to 3 | Factor CNT2<br>(Register interface (page 9-4)) | REG_CNT2_MUL (page 9-8) | 0×4A |

### Object 5827hex – Encoder Measuring Time Out

The object $5827_{hex}$ defines the time out (in 10 ms/ bit), after which a message (Object 5803hex (page 11-7) MSG_CNTx_FQE) is generated in period duration measurement.

*Table 11-14: Object $5827_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Measuring Time Out |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Structure of the data bytes**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | time out CNT1 in 10 ms/ bit (Register interface (page 9-4)) | REG_CNT1_TO (page 9-7) | 0×2D |
| **n + 1** = CNT2 | 0 to 3 | time out CNT2 in 10 ms/ bit (Register interface (page 9-4)) | REG_CNT2_TO (page 9-8) | 0×4D |

### Object 5830hex – Encoder Measuring Value

The object $5830_{hex}$ reads the measured value of the counters CNTx.

*Table 11-15: Object $5830_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Measuring Value |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | measured value CNT1 (Register interface (page 9-4)) | REG_CNT1_MV (page 9-5) | 0×21 |
| **n + 1** = CNT2 | 0 to 3 | Measured value CNT2 (Register interface (page 9-4)) | REG_CNT2_MV (page 9-7) | 0×41 |

### Object 5831hex – Encoder Latch Value

The object $5831_{hex}$ reads the value of the latch register for the counter CNTx.

*Table 11-16: Object $5831_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Encoder Latch Value |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | buffer CNT1<br>(Register interface (page 9-4)) | REG_CNT1_LATCH<br>(page 9-6) | 0×28 |
| **n + 1** = CNT2 | 0 to 3 | buffer CNT2<br>(Register interface (page 9-4)) | REG_CNT2_LATCH<br>(page 9-8) | 0×48 |

### Object 5901hex – PWM Config

The object $5901_{hex}$ affects the configuration parameters of PWMx:

Write accesses initiate a parameter update via the internal module bus of the BL20 station. The parameter is stored as a non-volatile parameter in the BL20 gateway and is restored with every node reset.

*Table 11-17: Object $5901_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Config |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≤ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = PWM1 | 0 | diagnostic PWM1 | X | mode D1 (0×3F) | | | | | |
| | 1 | **Bit 15** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | DBP1 STS MODE (0×00) | | substitute value P1 (0×00) | substitute value D1 (0×00) | mode PWM1 (0×00) | | | |
| | 2 | **Bit 23** | **Bit 22** | **Bit 21** | **Bit 20** | **Bit 19** | **Bit 18** | **Bit 17** | **Bit 16** |
| | | X | | | | | | | |
| | 3 | **Bit 31 (msb)** | **Bit 30** | **Bit 29** | **Bit 28** | **Bit 27** | **Bit 26** | **Bit 25** | **Bit 24** |
| | | X | ADR AUX REG1 WR DATA (0×60) | | | | | | |

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n + 1** = PWM2 | 0 | diagnostic PWM2 | X | mode D2 (0×3F) | | | | | |
| | 1 | **Bit 15** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | DBP2 STS MODE (0×00) | | substitute value P2 (0×00) | substitute value D2 (0×00) | mode PWM2 (0×00) | | | |
| | 2 | **Bit 23** | **Bit 22** | **Bit 21** | **Bit 20** | **Bit 19** | **Bit 18** | **Bit 17** | **Bit 16** |
| | | X | | | | | | | |
| | 3 | **Bit 31 (msb)** | **Bit 30** | **Bit 29** | **Bit 28** | **Bit 27** | **Bit 26** | **Bit 25** | **Bit 24** |
| | | X | ADR AUX REG3 WR DATA (0×70) | | | | | | |

( ) = default parameterization

**Note**

A more detailed description of the module's parameter data can be found in the field bus independent module description (chapter 3, section Parameter data of the module (page 3-10)).

### Object 5902hex – PWM Status

The object $5902_{hex}$ supplies the following status messages:

- output status
- operation status of the PWM

*Table 11-18: Object $5902_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Status |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data byte

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| **n** = PWM1 | 0 | X | X | STS_ DBP1 | D1 | STS_ PWM1_ LOGMSG | STS_ PWM1_ SFKT_EN | STS_ PWM1_ RUN | STS_ PWM1_ GENERAL_ EN |
| **n + 1** = PWM2 | 0 | X | X | STS_ DBP2 | D2 | STS_ PWM2_ LOGMSG | STS_ PWM2_ SFKT_EN | STS_ PWM2_ RUN | STS_ PWM2_ GENERAL_ EN |

**Note**

A more detailed description of the module's messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 5903hex– PWM Flags

The object $5903_{hex}$ supplies the following status messages:

- execution of a SW latch retrigger
- zero-crossing of the counter for the signal output
- an event defined as special function (SFKT) has occurred
- PWM output error

*Table 11-19: Object $5903_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Flags |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Access PDO | 0×01 to 0×47 | r |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Note**

By means of this access, the messages are reset automatically after reading.

### Structure of the data byte

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| **n** = PWM1 | 0 | X | X | X | X | MSG_ PWM1_ DO_ERR | MSG_ PWM1_ SFKT | MSG_ PWM1_ NDDC | MSG_ PWM1_ SW_LR |
| **n + 1** = PWM2 | 0 | X | X | X | X | MSG_ PWM2_ DO_ERR | MSG_ PWM2_ SFKT | MSG_ PWM2_ NDDC | MSG_ PWM2_ SW_LR |

**Note**

A more detailed description of the module's messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 5904hex – PWM Diag

The object $5904_{hex}$ reads the module's diagnostic byte.

*Table 11-20: Object $5904_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Diag |
| Object code | | ARRAY |
| PDO mapping | | – |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Note**

By means of this access, the messages are reset automatically after reading.

### Structure of the data byte

The meaning of the diagnostic bits generally corresponds to the general description of the module (→ see Diagnostic data of the module).

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| **n** = PWM1 | 0 | HW_ERR | PWM1_ PAR _ERR | X | X | X | X | P1_DIAG | D1_DIAG |
| **n + 1** = PWM2 | 0 | HW_ERR | PWM2_ PAR _ERR | X | X | X | X | P2_DIAG | D2_DIAG |

**Note**

A more detailed description of the module's diagnostic data can be found in the field bus independent module description (chapter 3, section Diagnostic data of the module (page 3-9)).

### Object 5908hex – PWM Control

The object $5908_{hex}$ provides the following control functions for the PWMx:

- general enabling or disabling of the PWM function
- start/ stop of the signal output
- storing of Error messages of the module (page 8-2) for readout without data loss
- enabling or disabling of the special function
- executing a software latch retrigger
- Setting the single/ continuous signal output

*Table 11-21: Object $5908_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Control |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 16 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Access PDO | 0×01 to 0×47 | r |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Note**

By means of this access, the messages are reset automatically after reading.

### Structure of the data byte

The meaning of the diagnostic bits generally corresponds to the general description of the module (→ see Diagnostic data of the module).

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = PWM1 | 0 | X | PWM1_ SINGLE | PWM1_ SW_LR | PWM1_ SFKT_ DISABLE | X | PWM1_ LOGMSG | PWM1_ ENABLE | PWM1_ GENERAL_ DISABLE |
| | 1 | **Bit 15 (msb)** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | X | X | X | X | X | X | SET_P1 | SET_D1 |

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = PWM1 | 0 | X | PWM1_ SINGLE | PWM1_ SW_LR | PWM1_ SFKT_ DISABLE | X | PWM1_ LOGMSG | PWM1_ ENABLE | PWM1_ GENERAL_ DISABLE |
| | 1 | **Bit 15 (msb)** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | X | X | X | X | X | X | SET_P1 | SET_D1 |
| **n + 1** = PWM2 | 0 | X | PWM2_ SINGLE | PWM2_ SW_LR | PWM2_ SFKT_ DISABLE | X | PWM2_ LOGMSG | PWM2_ ENABLE | PWM2_ GENERAL_ DISABLE |
| | 1 | **Bit 15 (msb)** | **Bit 14** | **Bit 13** | **Bit 12** | **Bit 11** | **Bit 10** | **Bit 9** | **Bit 8** |
| | | X | X | X | X | X | X | SET_P2 | SET_D2 |

**Note**

A more detailed description of the module's control bits can be found in the field bus independent module description (chapter 3, section Process output/ control interface (page 3-19)).

### Object 5910hex – PWM Load Prepare Value

The object $5910_{hex}$ contains the load value (load value register, PWM1 no. 0×64, PWM2 no. 0×74) for the "prepared loading" of the counters. Setting the count value of the signals to be given out to this value is event-driven.

*Table 11-22: Object $5910_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Load Prepare Value |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Integer 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = PWM1 | 0 to 3 | load value register of PWM1 (Register interface (page 9-4)) | REG_PWM1_CNTSV (page 9-9) | 0×64 |
| **n + 1** = PWM2 | 0 to 3 | load value register of PWM2 (Register interface (page 9-4)) | REG_PWM2_CNTSV (page 9-10) | 0×74 |

### Object 5913hex – PWM Duty Cycle

The object $5913_{hex}$ is used to set the duty cycle of the PWM pulse.

*Table 11-23: Object $5913_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Duty Cycle |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Integer 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Structure of the data bytes**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = PWM1 | 0 to 3 | Mark-to-space ratio PWM1, (Register interface (page 9-4)) | REG_PWM1_DC (page 9-9) | 0×61 |
| **n + 1** = PWM2 | 0 to 3 | Mark-to-space ratio PWM2, (Register interface (page 9-4)) | REG_PWM2_DC (page 9-10) | 0×71 |

**Note**

See also, chapter 5, Period Duration / Duty Cycle Definition (page 5-3).

### Object 5920hex – PWM Period Duration

The object $5920_{hex}$ contains the value for the period duration of the PWM.

*Table 11-24: Object $5920_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Period Duration |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Integer 32 |
| Access SDO | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Access PDO | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = PWM1 | 0 to 3 | Period duration of PWM1 (Register interface (page 9-4)) | REG_PWM1_PD (page 9-9) | 0×60 |
| **n + 1** = PWM2 | 0 to 3 | Period duration of PWM2 (Register interface (page 9-4)) | REG_PWM2_PD (page 9-10) | 0×70 |

**Note**

See also, chapter 5, Period Duration / Duty Cycle Definition (page 5-3).

### Object 5931hex – PWM Latch Value

The object $5931_{hex}$ reads the value of the latch register for the PWMx.

*Table 11-25: Object $5931_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | PWM Latch Value |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = PWM1 | 0 to 3 | Buffer PWM1<br>(Register interface (page 9-4)) | REG_PWM1_LATCH<br>(page 9-9) | 0×66 |
| **n + 1** = PWM2 | 0 to 3 | Buffer PWM2<br>(Register interface (page 9-4)) | REG_PWM2_LATCH<br>(page 9-10) | 0×76 |

### Object 6810hex – Preset Value For Multi-Sensor Devices

This object is used to adapt the encoder zero-point to the mechanical zero-point of the system. The object 6810hex (corresponds to object 6010hex in accordance with CiA DS-406) directly writes the counter value for the counter.

*Table 11-26: Object 6810$_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Preset Value For Multi-Sensor Devices |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | Actual binary value of the CNT1 (Register interface (page 9-4)) | REG_CNT2_CNT (page 9-7) | 0×20 |
| **n + 1** = CNT2 | 0 to 3 | Actual binary value of the CNT2 (Register interface (page 9-4)) | REG_CNT2_CNT (page 9-7) | 0×40 |

### Object 6820hex – Position Value For Multi-Sensor Devices

The object 6820hex (corresponds to object 6020hex in accordance with CiA DS-406) contains the counter's counter value. .

*Table 11-27: Object 6820$_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Preset Value For Multi-Sensor Devices |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | Actual binary value of the CNT1 (Register interface (page 9-4)) | REG_CNT2_CNT (page 9-7) | 0×20 |
| **n + 1** = CNT2 | 0 to 3 | Actual binary value of the CNT2 (Register interface (page 9-4)) | REG_CNT2_CNT (page 9-7) | 0×40 |

### Object 6B00hex – CAM 1 State Register

The object $6B00_{hex}$ (corresponds to object $6300_{hex}$ in accordance with CiA DS-406) indicates whether the counter status is within a specified range. This range is limited by CAM1 Low Limit (object $6B10_{hex}$) and CAM1 High Limit (object $6B20_{hex}$).

*Table 11-28: Object $6B00_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | CAM 1 State Register |
| Object code | | ARRAY |
| PDO mapping | | ✓ |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

The following applies:

STS_DBPx = 1

at (REG_CNTx_CMP0) ≤ (REG_CNTx_CNT) < (REG_CNTx_CMP1)

**Structure of the data byte**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | X | X | X | X | X | X | X | STAT_CAM1(STS_DBP1) |
| **n + 1** = CNT2 | 0 | X | X | X | X | X | X | X | STAT_CAM2(STS_DBP2) |

**Note**

The function is only given, if DBPx STS MODE = 00 is parameterized (→ see Parameter data of the module (page 3-10)).

### Object 6B01hex – CAM 1 Enable Register

The object $6B01_{hex}$ activates or deactivates the status message concerning the comparison result (object Object 6B00hex – CAM 1 State Register).

*Table 11-29: Object $6B01_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | CAM 1 Enable Register |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

Function setting for the special functions of the outputs.

**Structure of the data byte**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | X | X | X | X | X | X | X | EN_CAM1 |
| **n + 1** = CNT2 | 0 | X | X | X | X | X | X | X | EN_CAM2 |

### Object 6B02hex – CAM1 Polarity Register

The object $6B02_{hex}$ can invert the status message for the comparison result (Object 6B00hex – CAM 1 State Register).

*Table 11-30: Object $6B02_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | CAM1 Polarity Register |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

Function setting for the special functions of the outputs.

**Structure of the data byte**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | X | X | X | X | X | X | X | POL_ CAM1 |
| **n + 1** = CNT2 | 0 | X | X | X | X | X | X | X | POL_ CAM2 |

### Object 6B10hex – CAM 1 Low Limit

The object $6B10_{hex}$ contains the compare value CMP0 of the counter module (→ see also Function of the CMPx compare registers (page 4-5)).

The object corresponds to object $6310_{hex}$ in accordance with CiA DS-406.

*Table 11-31: Object $6B10_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | CAM 1 Low Limit |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | compare value 0 of CNT1 (Register interface (page 9-4)) | REG_CNT1_CMP0 (page 9-6) | 0×26 |
| **n + 1** = CNT2 | 0 to 3 | compare value 0 of CNT2 (Register interface (page 9-4)) | REG_CNT2_CMP0 (page 9-8) | 0×46 |

**Note**

If a compare value is loaded which is outside the count limits, the value is transferred and an error message is generated in REG_CONFIG_ERRSTS (Error messages in REG_CONFIG_ERRSTS for the count operation mode (CNT1 and CNT2) (page 7-3)).

### Object 6B20hex – CAM 1 High Limit

The object $6B2_{hex}$ contains the compare value CMP1of the counter module (→ ee also Function of the CMPx compare registers (page 4-5)).

The object corresponds to object $6320_{hex}$ in accordance with CiA DS-406.

*Table 11-32: Object $6B20_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | CAM 1 High Limit |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | compare value 1 of CNT1 (Register interface (page 9-4)) | REG_CNT1_CMP1 (page 9-6) | 0×27 |
| **n + 1** = CNT2 | 0 to 3 | compare value 1 of CNT2 (Register interface (page 9-4)) | REG_CNT2_CMP1 (page 9-8) | 0×47 |

**Note**

If a compare value is loaded which is outside the count limits, the value is transferred and an error message is generated in REG_CONFIG_ERRSTS (Error messages in REG_CONFIG_ERRSTS for the count operation mode (CNT1 and CNT2) (page 7-3)).

### Object 6B30hex – CAM 1 Hysteresis

The object $6B30_{hex}$ defines the hysteresis value for compare value CMP0 and CMP1 which affects the digital output Dx assigned to the counter and the STS_DBPx.

The object corresponds to object $6330_{hex}$ in accordance with CiA DS-406.

**Note**

See chapter 6, Special functions of the outputs Dx: Pulse output defined by a hysteresis (page 6-5).

*Table 11-33: Object $6B31_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | CAM 1 Hysteresis |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 16 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

**Structure of the data bytes**

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 1 | hysteresis of the DO and STS_DBP1 for CNT1 (Register interface (page 9-4)) | REG_CNT1_DO1_HYS (page 9-7) | 0×2F |
| **n + 1** = CNT2 | 0 to 1 | hysteresis of the D2 and STS_DBP2 for CNT2 (Register interface (page 9-4)) | REG_CNT2_DO1_HYS (page 9-8) | 0×4F |

### Object 6C00hex – Area State Register

Object $6C00_{hex}$ contains two status bits that indicate the count value falling below the lower count limit (Object 6C01hex – Work Area Low Limit (page 11-39)) and exceeding the upper count limit (Object 6C02hex – Work Area High Limit (page 11-40)).

The status bits are non-volatile. All status messages are reset by writing Object 5803hex – Encoder Flags (page 11-7) with any value. Object $5803_{hex}$ contains the redundant status information.

The object corresponds to object $6400_{hex}$ in accordance with CiA DS-406. .

**Note**

See chapter 6, Special functions of the outputs Dx: Pulse output defined by a hysteresis (page 6-5).

*Table 11-34: Object $6C00_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Area State Register |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 8 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | ro |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data byte

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 (lsb) |
|---|---|---|---|---|---|---|---|---|---|
| **n** = CNT1 | 0 | X | X | X | X | X | MSG_ CNT1_ UFLW | MSG_ CNT1_ OFLW | X |
| **n + 1** = CNT2 | 0 | X | X | X | X | X | MSG_ CNT2_ UFLW | MSG_ CNT2_ OFLW | X |

**Note**

A more detailed description of the module's messages can be found in the field bus independent module description (chapter 3, section Process input/ check-back interface (page 3-14)).

### Object 6C01hex – Work Area Low Limit

The object 6C01 $_{hex}$ defines the value for the lower count limit (→ see also Set count limits (page 4-3)).

In the event of an underflow, bit 2 in Object 6C00hex – Area State Register (page 11-38) and bit 6 in Object 5803hex – Encoder Flags (page 11-7) are set.

The object corresponds to object 6401$_{hex}$ in accordance with CiA DS-406.

*Table 11-35: Object 6C01$_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Work Area Low Limit |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | lower count limit of CNT1 (Register interface (page 9-4)) | REG_CNT1_LOLIMIT (page 9-6) | 0×24 |
| **n + 1** = CNT2 | 0 to 3 | lower count limit of CNT2 (Register interface (page 9-4)) | REG_CNT2_LOLIMIT (page 9-8) | 0×44 |

**Note**

If the count limits are loaded with ((REG_CNTx_HILIMIT) <= (REG_CNTx_LOLIMIT), the value is transferred and an error message is recorded in the REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

If a count limit is set so that the actual count value is outside the count range, the current value is set to the count limit which is closest to it. In this case *no* error message is recorded in register REG_CONFIG_ERRSTS.

### Object 6C02hex – Work Area High Limit

The object 6C02 $_{hex}$ defines the value for the uppe count limit (→ se also Set count limits (page 4-3)).

In the event of an underflow, bit 1 in Object 6C00hex – Area State Register (page 11-38) and bit 5 in Object 5803hex – Encoder Flags (page 11-7) are set.

The object corresponds to object 6402$_{hex}$ in accordance with CiA DS-406.

*Table 11-36: Object 6C02$_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Work Area High Limit |
| Object code | | ARRAY |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 8 |
| | 0×01 to 0×47 | Unsigned 32 |
| Access | 0×00 | ro |
| | 0×01 to 0×47 | rw |
| Default value | 0×00 | - |
| | 0×01 to 0×47 | - |

### Structure of the data bytes

| Sub-index<br>0×00 ≤ **n** ≥ 0×47 | Byte | Description | register | Register- no. |
|---|---|---|---|---|
| **n** = CNT1 | 0 to 3 | upper count limit of CNT1 (Register interface (page 9-4)) | REG_CNT1_HILIMIT (page 9-6) | 0×25 |
| **n + 1** = CNT2 | 0 to 3 | upper count limit of CNT2 (Register interface (page 9-4)) | REG_CNT2_HILIMIT (page 9-8) | 0×45 |

**Note**

If the count limits are loaded with ((REG_CNTx_HILIMIT) <= (REG_CNTx_LOLIMIT), the value is transferred and an error message is recorded in the REG_CONFIG_ ERRSTS (CNTx) (page 7-3).

If a count limit is set so that the actual count value is outside the count range, the current value is set to the count limit which is closest to it. In this case *no* error message is recorded in register REG_CONFIG_ERRSTS.

### Object 6FFFhex – Device Type

The object 6FFF $_{hex}$ specifies the type of the second device profile supported.

The object contains the value 0× 00 0A 01 96.

The low word (0×01 96 = $406_{dec}$) specifies the device profile.

The high word (0×00 0A describes the encoder type in accordance with CiA DS-406 ($10_{dec}$ = Multi Sensor Encoder Interface).

The object corresponds to object $67FF_{hex}$ in accordance with CiA DS-406.

*Table 11-37: Object 6FFF$_{hex}$*

| Feature | Sub-index | Description |
|---|---|---|
| Name | | Device Type |
| Object code | | VAR |
| PDO mapping | | - |
| Data type | 0×00 | Unsigned 32 |
| Access | 0×00 | ro |

## 11.2 Emergencies of the BL20-E-2CNT-2PWM

With the EMERGENCY $7010_{hex}$, a "general module error in the station" is reported.

Additionally EMERGENCY $7011_{hex}$ defines precisely, if a change in the module's diagnostic data occurs. This EMERGENCY is sent with every change in module's diagnostic data.

| Name nung | Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| | | Error code | | Error register | Additional information | | | | |
| **General module error** General module error | | $7010_{hex}$ | | Bit 0 =1 Bit 7 = 1 (→ see Table 11-38:) | 0 | 0 | 0 | 0 | 0 |
| **Change of Dia.** (change in byte 0 to 3 of the diagnostic data) | | $7011_{hex}$ | | | Module no. | 0 | 0 | 0 | 0 |

### 11.2.1 Structure of the emergency frames

BL20 CANopen supports Emergency Frames (EMCY) as standardized in CiA DS-301.

The COB-IDs for the EMCY telegrams are defined by the Predefined Master-Slave Connection Set: COB-ID = 129 - 1 + Node-ID

In the event of a communication error, not only the Emergency Error Code but also the Error register and additional information will be transmitted, so that the error can be more precisely identified. Only a part of the 5 bytes is used for the additional information. The remaining bytes are then 0.

**error register**

*Table 11-38: Bit assignment of BL20 Error registers*

| error register | M/O | Meaning |
|---|---|---|
| Bit 0 | M **A** | Generate the error message |
| Bit 1 | O **B** | current error |
| Bit 2 | O | voltage error |
| Bit 3 | O | temperature error |
| Bit 4 | O | communication error (overrun, error state) |
| Bit 5 | O | device-profile-specific error |
| Bit 6 | O | reserved |
| Bit 7 | O | manufacturer-specific error |

**A** *M = mandatory*
**B** *O = optional*

# 12 Representation of the BL20-E-2CNT-2PWM in EtherNet/IP™

## 12.1 Process data of the module in EtherNet/IP™

**Note**

The structure of the process data bits in EtherNet/IP™ basically corresponds to the general structure of the process data (→ see chapter 3, section Process data of the module (page 3-14)).

## 12.2 Vendor Specific Class for BL20-E-2CNT/2PWM Module Class (VSC126)

This class contains all information and parameters for the module BL20-E-2CNT/2PWM.

In this class parameter options are only deactivated if another parameter option is activated instead.

### 12.2.1 Class Instance of the VSC

The genreal Class Instance Attributes of the VSC are defined as follows:

*Table 12-1: Class instance*

| **Attr. no.** dec. (hex.) | **Attribute name** | **G**et/ **S**et | **typ.** | **Description** |
|---|---|---|---|---|
| 100 (64h) | Class revision | G | UINT | Contains the revision no. of the class. (maj. rel. *1000 + min. rel.). |
| 101 (65h) | Max. instance | G | USINT | Contains the number of the highest instance of a object which has been created on this level of the class hierarchy. |
| 102 (66h) | # of instances | G | USINT | Contains the number of object instances which have been created in this class. |
| 103 (67h) | Max. class attribute | G | USINT | Contains the number of the last class attribute which has been implemented. |

**Object instance**

*Table 12-2: Object instance*

| **Attr. no.** dec. (hex.) | **Attribute name** | **G**et/ **S**et | **typ.** | **Description** |
|---|---|---|---|---|
| 100 (64h) | Max object attribute | G | USINT | Contains the number of the last obejct attribute which has been implemented. |
| 101 (65h) | Module present | G | BOOL, | 0 = module missing, empty base module<br>1 = module is plugged |
| 102 (66h) | Terminal slot number | G | USINT | Contains the slot number of the module (1st base module right to the gateway = 1)<br>Corresponds to the respective instance number in the TERMINAL SLOT CLASS (see D301034). |
| 103 (67h) | Module ID | G | DWORD | Contains the module ID. |
| 104 (68h) | Module order number | G | UDINT | Contains the module's ident number. |
| 105 (69h) | Module order name | G | SHORT STRING | Contains the module name, e.g. "BL20-E-2CNT/2PWM". |
| 106 (6Ah) | Module revision number | G | USINT | Contains the version no. of the module's firmware. |
| 107 (6Bh) | Module type ID | G | ENUM USINT | Describes the module type:<br>– "1 (01h): digital I/O-module |

*Table 12-2: Object instance*

| **Attr. no.** dec. (hex.) | **Attribute name** | **G**et/ **S**et | **typ.** | **Description** |
|---|---|---|---|---|
| 108 (6Ch) | Module command interface | G/S | ARRAY | Control interface of the module<br>ARRAY OF: Control byte sequence |
| 109 (6Dh) | Module command interface | G | ARRAY | Check-back interface of the module<br>ARRAY OF: Check-back byte sequence |
| 110 (6Eh) | Diag Bit Count | G | USINT | Contains the number of diagnosis bytes, sent be the module. |
| 111 (6Fh) | Diag Bytes | G | ARRAY | Contains the module's diagnosis bytes. |
| 112 (70h) | Module registered index | G | ENUM USINT | Contains the index numbers from all module lists. |
| 113 (71h) to 116 (74h) | reserved | | | – |
| 117 (75h) | Input A1 | G/S | USINT | 0: Signal logic remains (LOW = 0 / HIGH = 1)**A**<br>1: Invert signal before processing |
| 118 (76h) | Input B1 | G/S | USINT | |
| 119 (77h) | Input Z1 | G/S | USINT | |
| 120 (78h) | diagnostic CNT1 | G/S | USINT | 0: Diagnostic messages of the function unit activated in diagnostic interface **A**<br>1: Diagnostic messages of the function unit deactivated in diagnostic interface A |
| 121 (79h) | measurement mode CNT1 | G/S | USINT | 0: Frequency measurement **A**<br>1: period duration measurement |
| 122 (7Ah) | main count direction CNT1 | G/S | USINT | 00: Basic function **A**<br>01: none<br>10: up<br>11: down |
| 123 (7Bh) | filter Z1 | G/S | USINT | 00: 2 µs **A**<br>01: 16 µs<br>10: reserved<br>11: reserved<br>Note<br>Irrespective of the setting for the filter property, the maximum input frequency of the channel has to be considered |
| 124 (7Ch) | filter A1, B1 | G/S | USINT | |
| 125 (7Dh) | Pull Up Z1 | G/S | USINT | 0: Pull Up resistance 20 kΩ off **A**<br>1: Pull Up resistance 20 kΩ on |

*Table 12-2: Object instance*

| Attr. no. dec. (hex.) | Attribute name | Get/ Set | typ. | Description |
|---|---|---|---|---|
| 126 (7Eh) | threshold input A,B,Z CNT1 | | USINT | 0: Threshold 7.5V (only valid for Ax, Bx, Zx) **A**<br>1: Threshold 2.5V (only valid for Ax, Bx, Zx) |
| 127 (7Fh) | mode Z1 (CNT1 page 4-12, PWM1 page 5-11) | | USINT | 0000: Alarm input CNT<br>0001: HW gate CNT **A**<br>0010: Single Latch-Retrigger CNT<br>0011: Continuous latch retrigger CNT<br>0100: Single L.-R. and HW gate CNT<br>0101: Continuous L.-R. and HW gate CNT<br>0110: reserved<br>0111: Alarm input PWM<br>1000: HW gate PWM<br>1001: Retrigger PWM<br>1010 to 1110 reseved<br>1111: Z just input |
| 128 (80h) | Mode CNTx (page 4-9) | | USINT | 0000: Pulse direction, single sample **A**<br>0001: Pulse direction, double sample<br>0010: AB mode, single sample<br>0011: AB mode, double sample<br>0100: AB mode, four samples<br>0101 to 1110 reserved<br>1111: AB just input |
| 129 (81h) | Input A2 | G/S | USINT | 0: Signal logic remains (LOW = 0 / HIGH = 1)**A**<br>1: Invert signal before processing |
| 130 (82h) | Input B2 | G/S | USINT | |
| 131 (83h) | Input Z2 | G/S | USINT | |
| 132 (84h) | diagnostics CNT2 | G/S | USINT | 0: Diagnostic messages of the function unit activated in diagnostic interface **A**<br>1: Diagnostic messages of the function unit deactivated in diagnostic interface A |
| 133 (85h) | measurement mode CNT2 | G/S | USINT | 0: Frequency measurement **A**<br>1: period duration measurement |
| 134 (86h) | main count direction CNT2 | G/S | USINT | 00: Basic function **A**<br>01: none<br>10: up<br>11: down |

*Table 12-2: Object instance*

| **Attr. no.** dec. (hex.) | **Attribute name** | **G**et/ **S**et | **typ.** | **Description** |
|---|---|---|---|---|
| 135 (87h) | filter Z2 | G/S | USINT | 00: 2 µs **A**<br>01: 16 µs<br>10: reserved<br>11: reserved<br>Note<br>Irrespective of the setting for the filter property, the maximum input frequency of the channel has to be considered |
| 136 (88h) | filter A2, B2 | G/S | USINT | |
| 137 (89h) | Pull Up Z2 | G/S | USINT | 0: Pull Up resistance 20 kΩ off **A**<br>1: Pull Up resistance 20 kΩ on |
| 138 (8Ah) | threshold input A,B,Z CNT2 | G/S | USINT | 0: Threshold 7.5V (only valid for Ax, Bx, Zx) **A**<br>1: Threshold 2.5V (only valid for Ax, Bx, Zx) |
| 139 (8Bh) | mode Z2 (CNT1 page 4-12, PWM1 page 5-11) | G/S | USINT | 0000: Alarm input CNT<br>0001: HW gate CNT **A**<br>0010: Single Latch-Retrigger CNT<br>0011: Continuous latch retrigger CNT<br>0100: Single L.-R. and HW gate CNT<br>0101: Continuous L.-R. and HW gate CNT<br>0110: reserved<br>0111: Alarm input PWM<br>1000: HW gate PWM<br>1001: Retrigger PWM<br>1010 to 1110 reseved<br>1111: Z just input |
| 140 (8Ch) | mode CNT2 (page 4-9) | G/S | USINT | 0000: Pulse direction, single sample **A**<br>0001: Pulse direction, double sample<br>0010: AB mode, single sample<br>0011: AB mode, double sample<br>0100: AB mode, four samples<br>0101 to<br>1110: reserved<br>1111: AB just input |
| 141 (8Dh) | diagnostics PWM1 | G/S | USINT | 0: Diagnostic messages of the function unit activated in diagnostic interface **A**<br>1: Diagnostic messages of the function unit deactivated in diagnostic interface A |
| 142 (8Eh) | mode D1 (page 6-2) | G/S | USINT | Definition of the function for Dx<br>(Default = 11 1111 → single output, can be controlled via process data) |

*Table 12-2: Object instance*

| Attr. no. dec. (hex.) | Attribute name | Get/ Set | typ. | Description |
|---|---|---|---|---|
| 143 (8Fh) | DBP1 STS MODE | G/S | USINT | – 00: STS_DBPx = 1 **A** at (REG_CNTx_CMP0) <= (REG_CNTx_CNT) < (REG_CNTx_CMP1)<br>– 01: reserved<br>– 10: reserved<br>– 11: STS_DBPx = Px |
| 144 (90h) | substitute value P1 | G/S | USINT | The output of the substitute value depends on the parameterization of the used gateway (→ documentation for the BL20-gateways). |
| 145 (91h) | substitute value D1 | G/S | USINT | |
| 146 (92h) | mode PWM1 (page 5-2) | G/S | USINT | 0000: PD DC Definition **A**<br>0001: HT LT Definition<br>0010 to 0111: reserved<br>1111: P just output |
| 147 (93h) | diagnostics PWM2 | G/S | USINT | 0: Diagnostic messages of the function unit activated in diagnostic interface **A**<br>1: Diagnostic messages of the function unit deactivated in diagnostic interface |
| 148 (94h) | mode D2 (page 6-2) | G/S | USINT | Definition of the function for Dx<br>(Default = 11 1111 → single output, can be controlled via process data) |
| 149 (95h) | DBP2 STS MODE | G/S | USINT | – 00: STS_DBPx = 1 **A** at (REG_CNTx_CMP0) <= (REG_CNTx_CNT) < (REG_CNTx_CMP1)<br>– 01: reserved<br>– 10: reserved<br>– 11: STS_DBPx = Px |
| 150 (96h) | substitute value P2 | G/S | USINT | The output of the substitute value depends on the parameterization of the used gateway (→ documentation for the BL20-gateways). |
| 151 (97h) | substitute value D2 | G/S | USINT | |
| 152 (98h) | mode PWM2 (page 5-2) | G/S | USINT | 0000: PD DC Definition **A**<br>0001: HT LT Definition<br>0010 to 0111: reserved<br>1111: P just output |
| 153 (99h) | ADR AUX REG1 RD DATA | G/S | USINT | Address of the basic write registers<br>(Default:<br>ADRAUXREG1WRDATA = 0x60,<br>ADRAUXREG2WRDATA = 0x61,<br>ADRAUXREG3WRDATA = 0x70) |
| 154 (9Ah) | ADR AUX REG1 RD DATA | G/S | USINT | |
| 155 (9Bh) | ADR AUX REG3 RD DATA | G/S | USINT | |

*Table 12-2: Object instance*

| Attr. no. dec. (hex.) | Attribute name | Get/ Set | typ. | Description |
|---|---|---|---|---|
| 156 (9Ch) | ADR AUX REG1 WR DATA | G/S | USINT | Address of the basic read registers (Default: ADR AUX REG1 RD DATA = 0x20, ADR AUX REG2 RD DATA = 0x21, ADR AUX REG3 RD DATA = 0x40) |
| 157 (9Dh) | ADR AUX REG2 WR DATA | G/S | USINT | |
| 158 (9Eh) | ADR AUX REG3 WR DATA | G/S | USINT | |

**Note**

A more detailed description of the module's parameter data can be found in the field bus independent module description (chapter 3, section Parameter data of the module (page 3-10)).

## 12.3 Process data of the module in EtherNet/IP™

### 12.3.1 Diagnostic data

The module's diagnostic data contain error messages that are operation and application relevant for the control system. 4 bytes are used to transfer the diagnostic data.

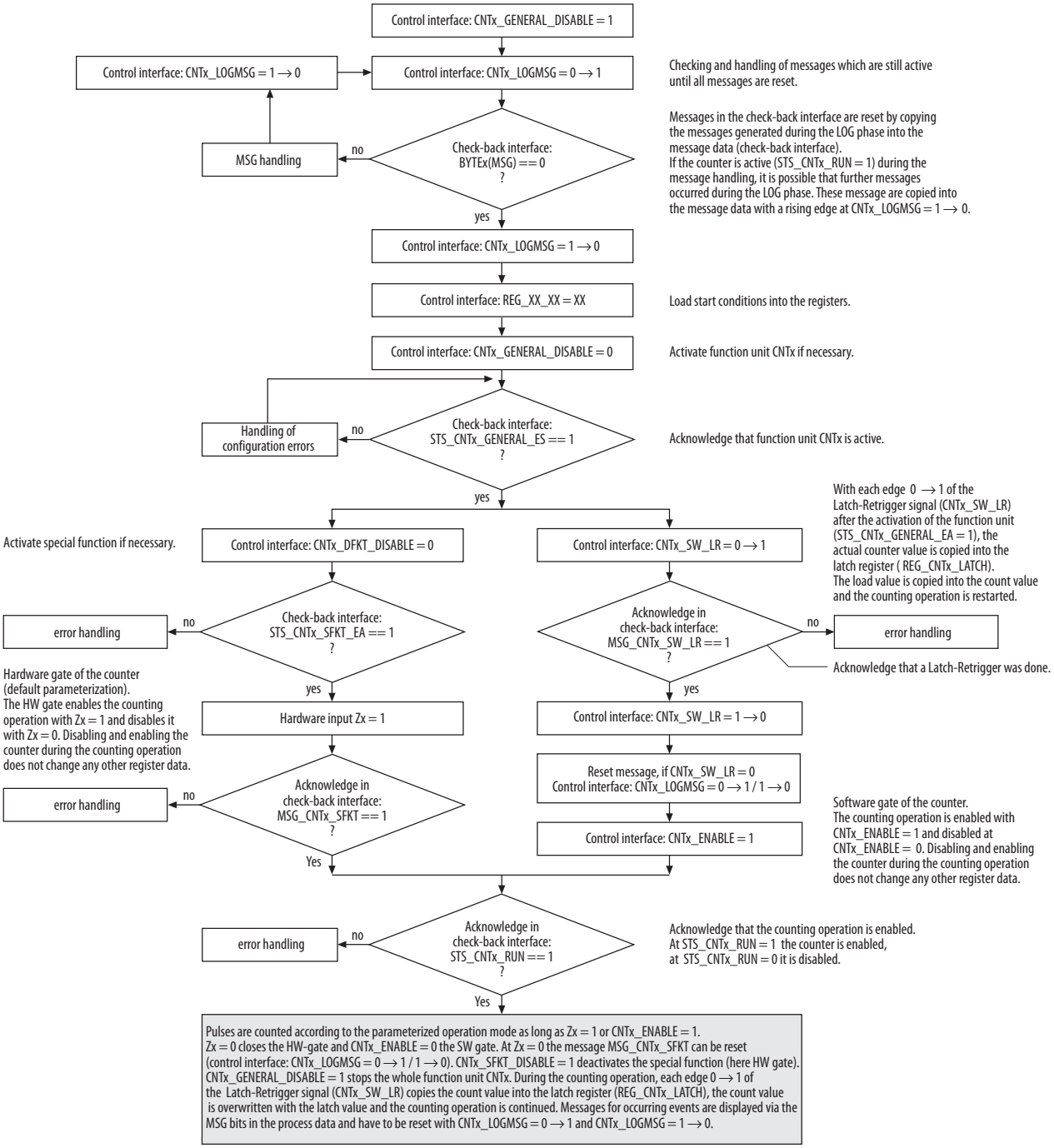
**Note**

A more detailed description of the module's diagnostic data can be found in the field bus independent module description (chapter 3, section Diagnostic data of the module (page 3-9)).

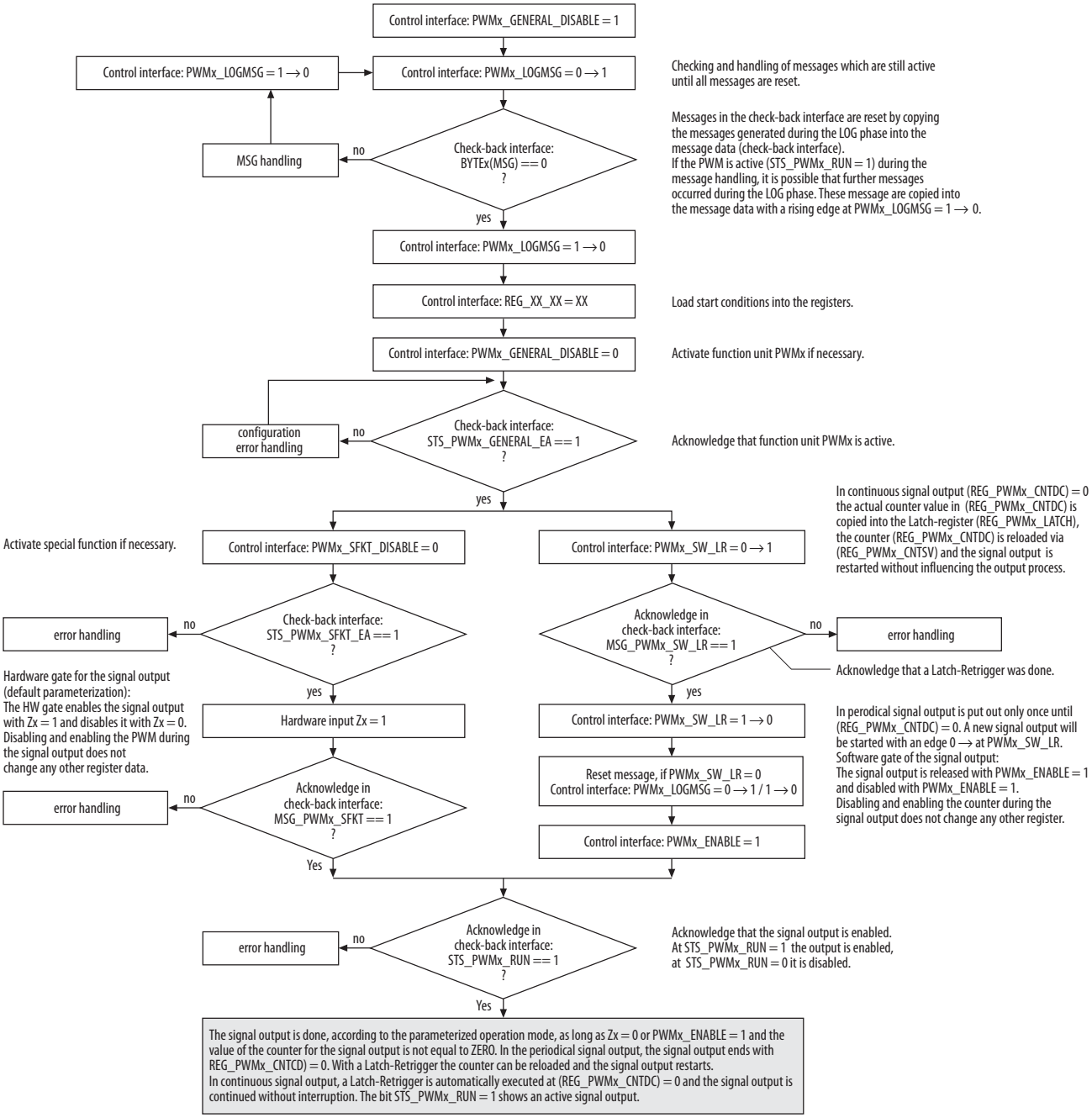
| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | HW_ERR | CNT1_ PAR_ERR | X | X | X | X | X | X |
| 1 | HW_ERR | CNT2_ PAR_ERR | X | X | X | X | X | X |
| 2 | HW_ERR | PWM1_ PAR_ERR | X | X | X | X | P1_DIAG | D1_DIAG |
| 3 | HW_ERR | PWM2_ PAR_ERR | X | X | X | X | P2_DIAG | D2_DIAG |

# 13 Appendix

## 13.1 Flow chart of a count operation



## 13.2 Flow chart of a pulse output



# 14 Index

## R



## S



## T



## U



## V



## W