

**WAGO** → **I/O** → **SYSTEM** 750

**Fieldbus Independent  
I/O Modules**

**RTC Module  
750-640**



**Manual**

Version 1.0.2

Copyright © 2006 by WAGO Kontakttechnik GmbH & Co. KG  
All rights reserved.

**WAGO Kontakttechnik GmbH & Co. KG**

Hansastraße 27  
D-32423 Minden

Phone: +49 (0) 571/8 87 – 0  
Fax: +49 (0) 571/8 87 – 1 69  
E-Mail: [info@wago.com](mailto:info@wago.com)  
Web: <http://www.wago.com>

**Technical Support**

Phone: +49 (0) 571/8 87 – 5 55  
Fax: +49 (0) 571/8 87 – 85 55  
E-Mail: [support@wago.com](mailto:support@wago.com)

Every conceivable measure has been taken to ensure the correctness and completeness of this documentation. However, as errors can never be fully excluded, we would appreciate any information or ideas at any time.

E-Mail: [documentation@wago.com](mailto:documentation@wago.com)

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally trademark or patent protected.

# Content

|   |           |
|---|-----------|
| <b>1 Important Comments .....</b>                                 | <b>5</b>  |
| 1.1 Legal Principles.....   | 5         |
| 1.1.1 Copyright .....   | 5         |
| 1.1.2 Personnel Qualification .....                               | 5         |
| 1.1.3 Intended Use .....  | 5         |
| 1.2 Symbols .....   | 6         |
| 1.3 Number Notation .....   | 6         |
| 1.4 Safety Notes .....  | 7         |
| 1.5 Scope .....   | 7         |
| <b>2 I/O Modules .....</b>  | <b>8</b>  |
| 2.1 Special Modules .....   | 8         |
| 2.1.1 750-640 [RTC Module] .....                                  | 8         |
| 2.1.1.1 View .....  | 8         |
| 2.1.1.2 Description.....  | 8         |
| 2.1.1.3 Display Elements .....                                    | 9         |
| 2.1.1.4 Schematic Diagram.....                                    | 10        |
| 2.1.1.5 Technical Data .....                                      | 11        |
| 2.1.1.6 Functional Description.....                               | 12        |
| 2.1.1.6.1 Date and Time .....                                     | 12        |
| 2.1.1.6.2 Configuration Data.....                                 | 13        |
| 2.1.1.6.2.1 Timer Configuration (T0 ... T31) .....                | 13        |
| 2.1.1.6.2.1.1 Example .....                                       | 14        |
| 2.1.1.6.2.2 Auto .....  | 15        |
| 2.1.1.6.2.3 TimeZone .....  | 15        |
| 2.1.1.6.2.4 DstBias .....   | 15        |
| 2.1.1.6.2.5 Password.....   | 15        |
| 2.1.1.6.2.6 Init.....   | 15        |
| 2.1.1.6.2.7 Antenna.....  | 16        |
| 2.1.1.6.3 Service Data .....                                      | 16        |
| 2.1.1.6.3.1 Hours Run and Maintenance Interval.....               | 17        |
| 2.1.1.7 Process Image .....                                       | 17        |
| 2.1.1.7.1 Structure of the Input Data and Output Data .....       | 18        |
| 2.1.1.7.2 Access to Module Data .....                             | 19        |
| 2.1.1.8 RTC Module Functions.....                                 | 20        |
| 2.1.1.8.1 Function Overview .....                                 | 20        |
| 2.1.1.8.2 Example of Function Call .....                          | 22        |
| 2.1.1.8.2.1 Data Reading .....                                    | 22        |
| 2.1.1.8.2.2 Data Receiving .....                                  | 22        |
| 2.1.1.8.2.3 Writing Data in Safe Configuration Area.....          | 23        |
| 2.1.1.9 Connection Examples .....                                 | 26        |
| 2.1.1.9.1 Antenna Supply with 24 V .....                          | 26        |
| 2.1.1.9.2 Antenna Supply with 3 V via External Power Supply ..... | 26        |
| <b>3 Attachment .....</b>   | <b>27</b> |
| 3.1 Functions .....   | 27        |

---

|          |  |    |
|----------|--|----|
| 3.1.1    | Function Overview .....                    | 27 |
| 3.1.2    | Function Overview sorted by ID .....       | 29 |
| 3.1.3    | Function Overview sorted by Function ..... | 31 |
| 3.1.4    | Function Reference .....                   | 33 |
| 3.1.4.1  | IDLE, 0x00 .....                           | 33 |
| 3.1.4.2  | GET_TOD100, 0x01 .....                     | 34 |
| 3.1.4.3  | GET_CURRENT_TOD100, 0x09 .....             | 35 |
| 3.1.4.4  | GET_TOD, 0x02 .....                        | 36 |
| 3.1.4.5  | GET_CURRENT_TOD, 0x0A .....                | 37 |
| 3.1.4.6  | GET_DT, 0x03 .....                         | 38 |
| 3.1.4.7  | GET_CURRENT_DT, 0x0B .....                 | 39 |
| 3.1.4.8  | GET_UTCTIME, 0x04 .....                    | 40 |
| 3.1.4.9  | GET_CURRENT_UTCTIME, 0x0C .....            | 41 |
| 3.1.4.10 | GET_DATE, 0x05 .....                       | 42 |
| 3.1.4.11 | GET_CURRENT_DATE, 0x0D .....               | 43 |
| 3.1.4.12 | GET_MS, 0x06 .....                         | 44 |
| 3.1.4.13 | GET_WEEKDAY, 0x07 .....                    | 45 |
| 3.1.4.14 | SET_DT, 0x0F .....                         | 46 |
| 3.1.4.15 | GET_CHANNEL, 0x10 .....                    | 47 |
| 3.1.4.16 | SET_CHANNEL, 0x11 .....                    | 48 |
| 3.1.4.17 | CLEAR_CHANNEL, 0x12 .....                  | 49 |
| 3.1.4.18 | GET_AUTO, 0x13 .....                       | 50 |
| 3.1.4.19 | SET_AUTO, 0x14 .....                       | 51 |
| 3.1.4.20 | GET_ADDR, 0x20 .....                       | 52 |
| 3.1.4.21 | SET_ADDR, 0x21 .....                       | 53 |
| 3.1.4.22 | GET_OPH, 0x30 .....                        | 54 |
| 3.1.4.23 | GET_OPH_INC (auto incremental), 0x31 ..... | 55 |
| 3.1.4.24 | SET_OPH, 0x32 .....                        | 56 |
| 3.1.4.25 | SET_OPH_INC (auto incremental), 0x33 ..... | 57 |
| 3.1.4.26 | GET_SERVICE_FLAGS, 0x34 .....              | 58 |
| 3.1.4.27 | GET_CNF, 0x38 .....                        | 59 |
| 3.1.4.28 | GET_CNF_INC (auto incremental), 0x39 ..... | 60 |
| 3.1.4.29 | SET_CNF, 0x3A .....                        | 61 |
| 3.1.4.30 | SET_CNF_INC (auto incremental), 0x3B ..... | 62 |
| 3.1.4.31 | GET_DIAG, 0xFF .....                       | 63 |

# 1 Important Comments

To ensure fast installation and start-up of the units described in this manual, we strongly recommend that the following information and explanations are carefully read and abided by.

## 1.1 Legal Principles

### 1.1.1 Copyright

This manual is copyrighted, together with all figures and illustrations contained therein. Any use of this manual which infringes the copyright provisions stipulated herein, is not permitted. Reproduction, translation and electronic and photo-technical archiving and amendments require the written consent of WAGO Kontakttechnik GmbH & Co. KG. Non-observance will entail the right of claims for damages.

WAGO Kontakttechnik GmbH & Co. KG reserves the right to perform modifications allowed by technical progress. In case of grant of a patent or legal protection of utility patents all rights are reserved by WAGO Kontakttechnik GmbH & Co. KG. Products of other manufacturers are always named without referring to patent rights. The existence of such rights can therefore not be ruled out.

### 1.1.2 Personnel Qualification

The use of the product detailed in this manual is exclusively geared to specialists having qualifications in PLC programming, electrical specialists or persons instructed by electrical specialists who are also familiar with the valid standards. WAGO Kontakttechnik GmbH & Co. KG declines all liability resulting from improper action and damage to WAGO products and third party products due to non-observance of the information contained in this manual.

### 1.1.3 Intended Use

For each individual application, the components supplied are to work with a dedicated hardware and software configuration. Modifications are only permitted within the framework of the possibilities documented in the manuals. All other changes to the hardware and/or software and the non-conforming use of the components entail the exclusion of liability on part of WAGO Kontakttechnik GmbH & Co. KG.

Please direct any requirements pertaining to a modified and/or new hardware or software configuration directly to WAGO Kontakttechnik GmbH & Co. KG.

## 1.2 Symbols

**Danger**

Always abide by this information to protect persons from injury.

**Warning**

Always abide by this information to prevent damage to the device.

**Attention**

Marginal conditions must always be observed to ensure smooth operation.

**ESD (Electrostatic Discharge)**

Warning of damage to the components by electrostatic discharge. Observe the precautionary measure for handling components at risk.

**Note**

Routines or advice for efficient use of the device and software optimization.

**More information**

References on additional literature, manuals, data sheets and INTERNET pages

## 1.3 Number Notation

| Number Code | Example              | Note                                    |
|-------------|----------------------|---|
| Decimal     | 100                  | normal notation                         |
| Hexadecimal | 0x64                 | C notation                              |
| Binary      | '100'<br>'0110.0100' | Within ',<br>Nibble separated with dots |

## 1.4 Safety Notes



### Warning

Switch off the system prior to working on bus modules!

In the event of deformed contacts, the module in question is to be replaced, as its functionality can no longer be ensured on a long-term basis.

The components are not resistant against materials having seeping and insulating properties. Belonging to this group of materials is: e.g. aerosols, silicones, triglycerides (found in some hand creams).

If it cannot be ruled out that these materials appear in the component environment, then additional measures are to be taken:

- installation of the components into an appropriate enclosure
  - handling of the components only with clean tools and materials.
- 



### Attention

Cleaning of soiled contacts may only be done with ethyl alcohol and leather cloths. Thereby, the ESD information is to be regarded.

Do not use any contact spray. The spray may impair the functioning of the contact area.

The WAGO-I/O-SYSTEM 750 and its components are an open system. It must only be assembled in housings, cabinets or in electrical operation rooms. Access must only be given via a key or tool to authorized qualified personnel.

The relevant valid and applicable standards and guidelines concerning the installation of switch boxes are to be observed.

---



### ESD (Electrostatic Discharge)

The modules are equipped with electronic components that may be destroyed by electrostatic discharge. When handling the modules, ensure that the environment (persons, workplace and packing) is well grounded. Avoid touching conductive components, e.g. gold contacts.

---

## 1.5 Scope

This manual describes the Special Module 750-640 RTC Module of the modular WAGO-I/O-SYSTEM 750.

Handling, assembly and start-up are described in the manual of the Fieldbus Coupler. Therefore this documentation is valid only in the connection with the appropriate manual.

## 2 I/O Modules

### 2.1 Special Modules

#### 2.1.1 750-640 [RTC Module]

Real Time Clock Module

##### 2.1.1.1 View

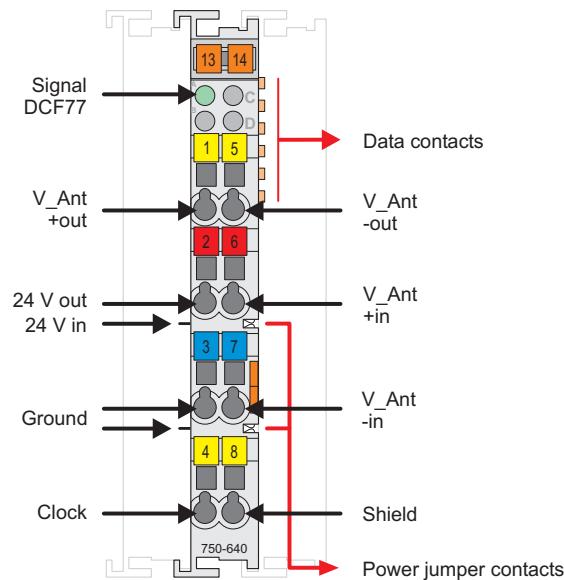


Fig. 2.1.1-1: View

g064000e

##### 2.1.1.2 Description

The RTC module 750-640 provides the higher-level control system with the actual time. The time continues to run in the event of a power failure. Once the module is switched on, about one hour is required for the integrated backup capacitor to be fully charged.

When an external antenna is connected, the clock can be set using the time signal from DCF77, WWVB or MSF. The antenna can be supplied directly via the module. Connecting an external antenna to operate the RTC module is not absolutely necessary.

With its 32 channels, the integrated time switch clock function makes it easier for the control unit to process time-triggered actions. The module also counts the power-on time of the 32 channels.

An external antenna is connected to both V\_ANT+ out and V\_ANT- out as well as to the clock.

Supplying the antenna is done via the V\_ANT+ in and V\_ANT- in terminals. Synchronizing the clock is done at the clock input using the time signal from DCF77 or WWVB. The module is set by default to receive the DCF77 signal.

The shield (screen) is directly connected to the DIN rail. A capacitive connection is made automatically when the module is snapped onto the DIN rail.

The status of the antenna input signal is indicated with a green LED.

An optocoupler is used for electrical isolation between the bus and the field side.

Any configuration of the specialty modules is possible when designing the fieldbus node. Grouping of module types is not necessary.

The field side supply voltage of 0 V and 24 V to downstream I/O modules is derived from adjacent I/O modules or from a supply module. The supply voltage for the field side is made automatically through the individual I/O modules by means of power jumper contacts.



### Warning

The maximum current of the internal power jumper contacts is 10 A. When configuring the system it is important not to exceed the maximum/sum current. However, if such a case should occur, another supply module must be added.

The RTC module 750-640 can be used with all couplers / controllers of the WAGO-I/O-SYSTEM 750 (except for the economy types 750-320, -323, -324 and -327).

#### 2.1.1.3 Display Elements

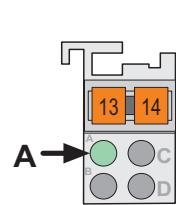


Fig. 2.1.1-2:  
Display Elements  
g063002x

| LED   | Description    | State              | Function   |
|-------|----------------|--------------------|--|
| green | Antenna signal | off                | No antenna signal  |
|       |                | Fast blinking      | No or faulty antenna signal  |
|       |                | Blinking with 1 Hz | Antenna signal is present, but not yet a correct telegram received |
|       |                | on                 | Antenna signal receiver is error free                              |

#### 2.1.1.4 Schematic Diagram

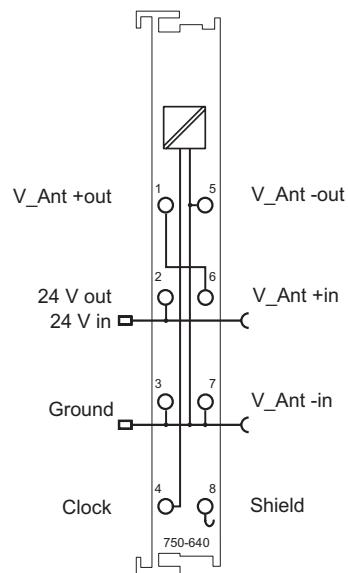


Fig. 2.1.1-3: Schematic Diagram

g064001e

### 2.1.1.5 Technical Data

| Module Specific Data  |  |
|---|--|
| Current consumption typ.<br>(internal)  | < 20 mA  |
| Voltage via power jumper contacts   | DC 24 V (-15%...+20%)  |
| Clock   |  |
| Accuracy (+25°C)  | < 1min/month   |
| Accuracy (+10...+40°C)  | < 2min/month   |
| Accuracy (-25...+85°C)  | < 7min/month   |
| Drift   | < 2min/year  |
| Buffer length   | > 6 days   |
| Clock Timer   |  |
| Number of channels  | 32   |
| Switching points  | 32 (per 32 channels on/off)  |
| DCF77 Input   |  |
| Signal voltage (0)  | -24 ... +1 V   |
| Signal voltage (1)  | 3 ... 24 V   |
| Idle run voltage  | DC 4 V   |
| Input filter  | 10 ms  |
| Input current typ.  | < 5 mA (at 24 V)<br>< 1 mA (at 5 V)  |
| Supply voltage V <sub>ant, in</sub>   | DC 5 ... 24 V  |
| Isolation   | 500 V (field / system)   |
| Input current typ. (field)  | 11 mA + load   |
| Internal data size  | 1 x 40 bit data (in/out) (5 byte user data)<br>1 x 8 bit control / status (optional) |
| Weight  | ca 55 g  |
| Approvals (cf. Chapter 2.2 of the Coupler/Controller Manual)                        |  |
|  | cUL <sub>US</sub> (UL508)  |
|  | Conformity Marking   |



### More Information

Detailed references to the approvals are listed in the document "Overview Approvals WAGO-I/O-SYSTEM 750", which you can find on the CD ROM ELECTRONICC Tools and Docs (Item-No.: 0888-0412) or in the internet under:

[www.wago.com](http://www.wago.com) → Documentation → WAGO-I/O-SYSTEM 750 → System Description

### 2.1.1.6 Functional Description

The RTC module provides many functions to read and set date and time, configuration data like time settings, time zone or antenna selection as well as service data such as run hours and maintenance intervals.

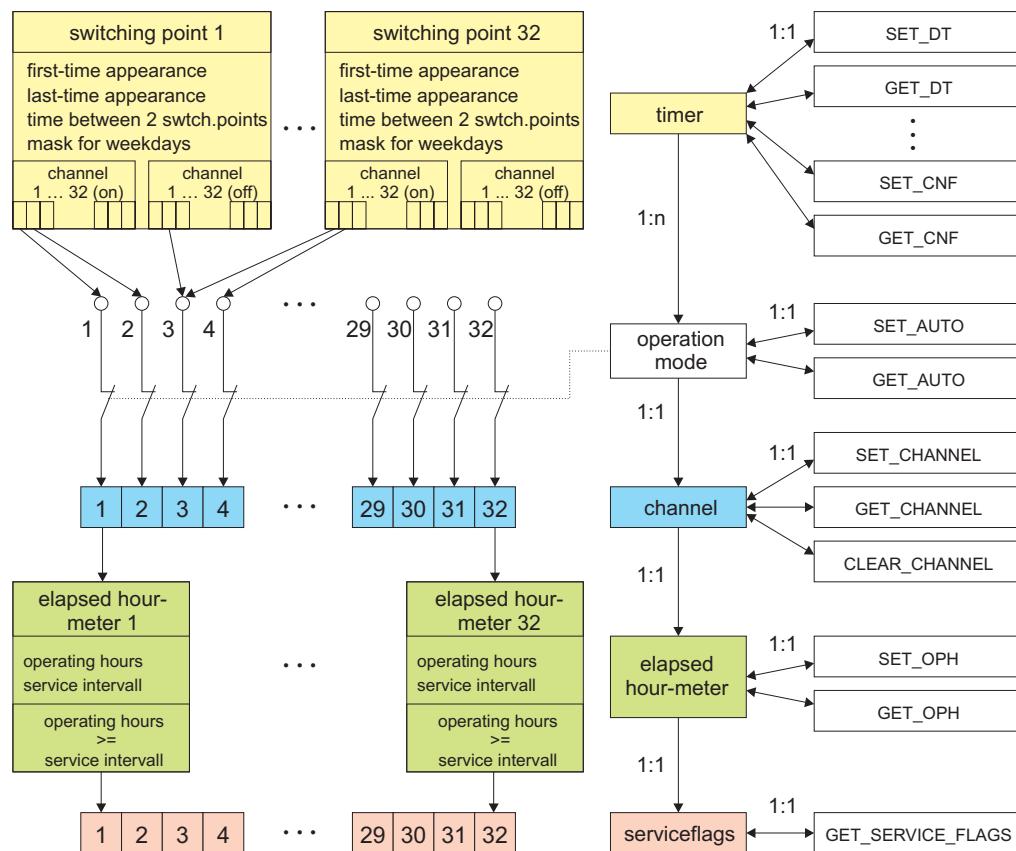


Fig. 2.1.1-4: Overview of RTC functions

g064005e

The call and the parameters of the individual functions are described in the attachment.

#### 2.1.1.6.1 Date and Time

The following functions are used to get the current date and time:  
GET\_TOD100, GET\_CURRENT\_TOD100, GET\_TOD,  
GET\_CURRENT\_TOD, GET\_DT, GET\_CURRENT\_DT, GET\_UTC,  
GET\_CURRENT\_UTC, GET\_DATE, GET\_CURRENT\_DATE, GET\_MS,  
GET\_WEEKDAY. The functions GET\_CURRENT\_.... are used to get the

data continuously.

Using the SET\_DT function, the date and time can be set manually.

### 2.1.1.6.2 Configuration Data

The configuration data of the RTC module can be accessed using the GET\_CNF, GET\_CNF\_INC, SET\_CNF and SET\_CNF\_INC functions. The address offset of the relevant entry must be previously set using the SET\_ADDR function. When the data is accessed via the GET\_CNF\_INC or SET\_CNF\_INC functions, the address offset is automatically increased by 4 bytes after the read or write operation is complete. For a write access with SET\_CNF and SET\_CNF\_INC, the valid password must previously be entered at address offset 0x0310.

The configuration data is structured as follows:

| Offset | Function | Description  | Length |
|--------|----------|--|--------|
| 0x0000 | T0       | Configuration Timer 0                                  | 24     |
| 0x0018 | T1       | Configuration Timer 1                                  | 24     |
| 0x0030 | T2       | Configuration Timer 2                                  | 24     |
| ...    | ...      | ...  | ...    |
| 0x02E8 | T31      | Configuration Timer 31                                 | 24     |
| 0x0300 | Auto     | Auto/manual for each channel (0 = auto)                | 4      |
| 0x0308 | TimeZone | Offset for time zone (seconds)                         | 4      |
| 0x030C | DstBias  | Offset for summer time (daylight saving time, seconds) | 4      |
| 0x0310 | Password | 0x3EA8D0F1   | 4      |
| 0x0314 | Init     | 0x1F0D8AE3   | 4      |
| 0x031C | Antenna  | Selection of the time signal transmitter               | 4      |

### 2.1.1.6.2.1 Timer Configuration (T0 ... T31)

The module has a timer function with 32 switching points and 32 output channels. Any of the 32 channels can be switched on or off at each switching point. All switching points are cyclically recurrent. The actual status of the switching channels can be queried via the GET\_CHANNEL function and can be changed manually via the SET\_CHANNEL and CLEAR\_CHANNEL functions.

A switching point is defined using the following entries:

| Offset | Meaning                           | Format  | Length | Comment          |
|--------|-----------------------------------|---|--------|------------------|
| 0      | First-time appearance             | DATE_AND_TIME   | 32 bit |                  |
| 4      | Last appearance                   | DATE_AND_TIME   | 32 bit | 0 = not used     |
| 8      | Time between two switching points | DATE_AND_TIME   | 32 bit | 0 = single event |
| 12     | Switched channels (on)            | Binary  | 32 bit | Dominant *)      |
| 16     | Switched channels (off)           | Binary  | 32 bit | Recessive        |
| 20     | Mask for weekdays                 | Bit 0 = Mo ...<br>Bit 6 = So,<br>0 = every,<br>0x7F = every | 8 bit  |                  |

\*) When a channel is accessed by two or more switching points, the point that is switching on is enforced.

With an entry in the configuration data, the operation mode can be switched into automatic or manual for each channel. The mode can also be selected via the SET\_AUTO function. When switching to the manual mode, the current status is pending until it is set manually again. When switching to the automatic mode, the current status (which may be set manually) is pending until it is set anew automatically.

After power restoration, the last occurrence of all timers as well as the last manual setting is taken into account, and the last theoretical status held during the powerless time is set for each channel.

#### 2.1.1.6.2.1.1 Example

Switching channels are used in an office for temperature and lighting control. Channel 1 controls a temperature regulator, which has a binary signal as an input, so that day and night time can be selected. On weekdays, the temperature is increased at 06:00 and reduced at 18:00.

The heating is turned completely off during vacation days from 26.05.2003 to 13.06.2003.

Room lighting is controlled by channel 4. The controller queries the light switch and the channel is set manually via SET\_CHANNEL and CLEAR\_CHANNEL. If the employees forget to switch off the light in the evening, this will be done automatically at 19:30.

Setting the timers is done as follows:

| <b>Timer</b> | <b>Start</b>           | <b>Stop</b>            | <b>Interval</b> | <b>On</b> | <b>Off</b> | <b>Days</b> |
|--------------|------------------------|------------------------|-----------------|-----------|------------|-------------|
| 1            | 01.01.1970<br>06:00:00 | -                      | 24h             | Channel 1 | -          | Mo-Fr       |
| 2            | 01.01.1970<br>18:00:00 | -                      | 24h             | -         | Channel 1  | Every       |
| 3            | 26.05.2003<br>06:00:01 | 13.06.2003<br>06:00:01 | 24h             | -         | Channel 1  | Every       |
| 4            | 01.01.1970<br>19:30:00 | -                      | 24h             | -         | Channel 4  | Every       |

The hexadecimal values are as follows:

| <b>Timer</b> | <b>Start</b> | <b>Stop</b> | <b>Interval</b> | <b>On</b>  | <b>Off</b> | <b>Days</b> |
|--------------|--------------|-------------|-----------------|------------|------------|-------------|
| 1            | 0x00005460   | 0x00000000  | 0x00015180      | 0x00000001 | 0x00000000 | 0x1F        |
| 2            | 0x0000FD20   | 0x00000000  | 0x00015180      | 0x00000000 | 0x00000001 | 0x00        |
| 3            | 0x3ED1AD61   | 0x3EE96861  | 0x00015180      | 0x00000000 | 0x00000001 | 0x00        |
| 4            | 0x00011238   | 0x00000000  | 0x00015180      | 0x00000000 | 0x00000008 | 0x00        |

#### 2.1.1.6.2.2 Auto

The automatic mode of the relevant switching channel is switched off for each bit set in Auto. The switching channel can only be changed using the SET\_CHANNEL and CLEAR\_CHANNEL functions.

#### 2.1.1.6.2.3 TimeZone

Time zone. For CET applies e.g. +1h = 3600s ≈ 0x00000E10.

#### 2.1.1.6.2.4 DstBias

For the transition from winter time to summer time, the content of DstBias (Daylight Saving Time) is added to the TimeZone when the time is being evaluated. This way, the transition can be performed without having to change the time or time zone.

#### 2.1.1.6.2.5 Password

Access to the configuration data is protected by means of a password. Therefore, only write access is possible if 0x3EA8D0F1 has been entered.

#### 2.1.1.6.2.6 Init

When 0x1FD8AE3 is written to Init, all configuration data is overwritten with zero.

### 2.1.1.6.2.7 Antenna

This is how to select the decoder scheme for time signal reception.

| Byte | Bit     | Function   |
|------|---------|--|
| 0    | 0 ... 1 | 0: No time signal reception<br>1: DCF77 Decoder activated<br>2: WWVB Decoder activated |
| 0    | 2       | DCF77 Polarity reversed<br>0 = Low-active<br>1 = High-active                           |

Note on setting the time zone and summer time using different antennas:

- The DCF77 transmitter sends CET (Central European Time) or CEST (Central European Summer Time), the summer time setting (DstBias) remains set to zero. The time zone for Germany is 0.
- The WWVB transmitter sends in UTC. If the module is operated at the US east coast (Eastern Time) for example, an offset of 5 hours (0xFFFFB9B0) must be entered in the TimeZone. The transition from winter time to summer time is done using a status bit within the time telegram. By setting this bit, one more hour will be transmitted by the module. Within the module, the transition is always performed at 00:00.

### 2.1.1.6.3 Service Data

Getting or setting the "up to now accumulated run time and maintenance interval" is done using the GET\_OPH, GET\_OPH\_INC, SET\_OPH and SET\_OPH\_INC functions for each channel. The address offset of the relevant hours-run meter must previously be set using the SET\_ADDR function. When the data is accessed via the GET\_CNF\_INC or SET\_CNF\_INC functions, the address offset is automatically increased by 4 bytes after the read or write operation is complete. For a write access with SET\_CNF and SET\_CNF\_INC, the valid password must previously be entered at address offset 0x0310 .

The hours-run meter is stopped in the event of a power failure independent of the status of the relevant channel.

| Offset | Function                   | Length |
|--------|----------------------------|--------|
| 0x0000 | Hours-run meter channel 0  | 8      |
| 0x0008 | Hours-run meter channel 1  | 8      |
| 0x0010 | Hours-run meter channel 2  | 8      |
| ...    | ...                        | ...    |
| 0x00F8 | Hours-run meter channel 31 | 8      |

### 2.1.1.6.3.1 Hours Run and Maintenance Interval

The hours run meter's data contains the "up to now accumulated hours run" as well as the set maintenance interval as shown below:

| Offset | Description                               | Length |
|--------|---|--------|
| 0x0000 | Up to now accumulated hours run (seconds) | 4      |
| 0x0004 | Maintenance interval (seconds)            | 4      |

The relevant service flag is set when the number of "up to now accumulated hours run" exceeds the specified maintenance interval for one switching channel. The GET\_SERVICE\_FLAGS function is used to get the status of the service flags.

### 2.1.1.7 Process Image

Using the RTC module 750-640, a 6-byte input and output process image maximum can be transferred to the fieldbus coupler/controller via one logical channel. The data received by the transmitter is stored in 4 input and output bytes (D0 ... D3). 1 control byte (C) and 1 status byte (S) are used to control the function call. 1 command byte (ID) is used to select and acknowledge the function to be executed.



#### Attention

The representation of the process data of some I/O modules or their variations in the process image depends on the fieldbus coupler/controller used. Please take this information as well as the particular design of the respective control/status bytes from the section "Fieldbus Specific Design of the Process Data" included in the description concerning the process image of the corresponding coupler/controller.

The input bytes constitute the memory area of the data to be transmitted by the module. The data to be transmitted to the module is written into the output bytes.

| Input Data |                       | Output Data |                       |
|------------|-----------------------|-------------|-----------------------|
| S          | Status byte           | C           | Control byte          |
| ID         | Function selection ID | ID          | Function selection ID |
| D0         | Data byte 0           | D0          | Control byte 0        |
| D1         | Data byte 1           | D1          | Control byte 1        |
| D2         | Data byte 2           | D2          | Control byte 2        |
| D3         | Data byte 3           | D3          | Control byte 3        |

The control byte includes the following bits:

| Control Byte |       |       |       |       |       |       |       |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| Bit 7        | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0            | 0     | 0     | 0     | 0     | 0     | 0     | TR    |

TR      Transmit Request, request to send (toggle flag)

0      reserved (constant must be set to zero)

The status byte contains the following bits:

| Status Byte |       |       |       |       |       |       |       |
|-------------|-------|-------|-------|-------|-------|-------|-------|
| Bit 7       | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |

TA      Transmit Acknowledge, transmission confirmation (toggle flag)

0      reserved (this value is always 0)

#### 2.1.1.7.1 Structure of the Input Data and Output Data

| Function Request |                          |       |       |       |       |       |       |       |
|------------------|--------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$                    | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0                        | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | Opcode                   |       |       |       |       |       |       |       |
| D0               | Request parameter byte 0 |       |       |       |       |       |       |       |
| D1               | Request parameter byte 1 |       |       |       |       |       |       |       |
| D2               | Request parameter byte 2 |       |       |       |       |       |       |       |
| D3               | Request parameter byte 3 |       |       |       |       |       |       |       |

0      reserved (value must always be 0).

TR      Transmit-Request    A changing is followed by a function call.  
(toggle flag)

Opcode    Instruction code of    A changing is followed by a function call.  
the function call

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | Opcode (mirrored)         |       |       |       |       |       |       |       |
| D0                | Response parameter byte 0 |       |       |       |       |       |       |       |
| D1                | Response parameter byte 1 |       |       |       |       |       |       |       |
| D2                | Response parameter byte 2 |       |       |       |       |       |       |       |
| D3                | Response parameter byte 3 |       |       |       |       |       |       |       |

X reserved (value is always 0 and is not evaluated)

TA Transmit-Acknowledge (toggle flag) A function call is received with a changing.

Opcode Mirrored instruction code of the function call A function call is received with a changing.

### 2.1.1.7.2 Access to Module Data

The user first writes the call parameters, and then the opcode and toggle flag. Once the opcode or toggle flag have changed, the output data is interpreted as complete function call by the RTC module so that the function can be executed. The function response is stored in the input data.

After the function has been executed, the opcode and toggle flags are mirrored by the RTC module. This way, the user knows that his request has been processed and the result is in the input data.

The toggle flag is necessary so that two function calls using the same opcode (even though different parameters may be used) can be performed one directly after the other.

## 2.1.1.8 RTC Module Functions

### 2.1.1.8.1 Function Overview

| ID                             | Function           | Description   | Site |
|--------------------------------|--------------------|---|------|
| 0x00                           | IDLE               | No function call  | 33   |
| <b>Date and Time Functions</b> |                    |   |      |
| 0x01                           | GET_TOD100         | Getting time in 100µs since 00:00                                   | 34   |
| 0x09                           | GET_CURRENT_TOD100 | Getting time in 100µs since 00:00, (consecutively)                  | 35   |
| 0x02                           | GET_TOD            | Getting time in milliseconds since 00:00                            | 36   |
| 0x0A                           | GET_CURRENT_TOD    | Getting time in milliseconds since 00:00 (consecutively)            | 37   |
| 0x03                           | GET_DT             | Getting time in seconds since 1.1.1970 00:00                        | 38   |
| 0x0B                           | GET_CURRENT_DT     | Getting time in seconds since 1.1.1970 00:00, (consecutively)       | 39   |
| 0x04                           | GET_UTC            | Getting time in seconds since 1.1.1970 00:00 (UTC)                  | 40   |
| 0x0C                           | GET_CURRENT_UTC    | Getting time in seconds since 1.1.1970 00:00 (UTC), (consecutively) | 41   |
| 0x05                           | GET_DATE           | Getting date in seconds since 1.1.1970 00:00                        | 42   |
| 0x0D                           | GET_CURRENT_DATE   | Getting date in seconds since 1.1.1970 00:00, (consecutively)       | 43   |
| 0x06                           | GET_MS             | Getting millisecond part of the last GET_DT request                 | 44   |
| 0x07                           | GET_WEEKDAY        | Getting weekday   | 45   |
| 0x0F                           | SET_DT             | Setting time in seconds since 1.1.1970 00:00                        | 46   |
| <b>Timer Functions</b>         |                    |   |      |
| 0x10                           | GET_CHANNEL        | Getting switching states of the switching channels                  | 47   |
| 0x11                           | SET_CHANNEL        | Setting switching states of the switching channels                  | 48   |
| 0x12                           | CLEAR_CHANNEL      | Clearing switching states of the switching channels                 | 49   |
| 0x13                           | GET_AUTO           | Getting automatic states of the switching channels                  | 50   |
| 0x14                           | SET_AUTO           | Setting automatic states of the switching channels                  | 51   |

| ID                                  | Function          | Description  | Site |
|-------------------------------------|-------------------|--|------|
| <b>Address Functions</b>            |                   |  |      |
| 0x20                                | GET_ADDR          | Getting the address offset for configuration data or hours-run meter | 52   |
| 0x21                                | SET_ADDR          | Setting address offset for configuration data or service data        | 53   |
| <b>Service Data Functions</b>       |                   |  |      |
| 0x30                                | GET_OPH           | Getting service data   | 54   |
| 0x31                                | GET_OPH_INC       | Getting service data, (auto incremental)                             | 55   |
| 0x32                                | SET_OPH           | Setting service data   | 56   |
| 0x33                                | SET_OPH_INC       | Setting service data, (auto incremental)                             | 57   |
| 0x34                                | GET_SERVICE_FLAGS | Getting service indication of the hours-run meter                    | 58   |
| <b>Configuration Data Functions</b> |                   |  |      |
| 0x38                                | GET_CNF           | Getting configuration data   | 59   |
| 0x39                                | GET_CNF_INC       | Getting configuration data, (auto incremental)                       | 60   |
| 0x3A                                | SET_CNF           | Setting configuration data   | 61   |
| 0x3B                                | SET_CNF_INC       | Setting configuration data, (auto incremental)                       | 62   |
| 0xFF                                | GET_DIAG          | Getting self-test diagnostics  | 63   |

### 2.1.1.8.2 Example of Function Call

#### 2.1.1.8.2.1 Data Reading

- Writing the data to be transmitted in output byte D0 ... D3
- Specifying the function with ID or reversing TR
- Transfer to the transmit buffer is complete when TA == TR

##### Example:

Write 0x12345678 as address offset (SET\_ADDR, ID 0x21)

Original values:

| Output data | C    | ID   | D0   | D1   | D2   | D3   |
|-------------|------|------|------|------|------|------|
| Value       | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |
| Input Data  | S    | ID   | D0   | D1   | D2   | D3   |
| Value       | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

Writing data:

| Output data | C    | ID   | D0   | D1   | D2   | D3   |
|-------------|------|------|------|------|------|------|
| Value       | 0x00 | 0x00 | 0x78 | 0x56 | 0x34 | 0x12 |
| Input Data  | S    | ID   | D0   | D1   | D2   | D3   |
| Value       | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

Writing ID (SET\_ADDR, ID 0x21) or reversing TR (Bit 0 in control byte C), data is transmitted:

(to make sure that the data is transmitted correctly, ID or TR should only be modified simultaneously or after setting D0...D3.)

| Output data | C    | ID   | D0   | D1   | D2   | D3   |
|-------------|------|------|------|------|------|------|
| Value       | 0x01 | 0x21 | 0x78 | 0x56 | 0x34 | 0x12 |
| Input Data  | S    | ID   | D0   | D1   | D2   | D3   |
| Value       | 0x01 | 0x21 | 0x00 | 0x00 | 0x00 | 0x00 |

| Output data | C    | ID   | D0   | D1   | D2   | D3   |
|-------------|------|------|------|------|------|------|
| Value       | 0x01 | 0x21 | 0x00 | 0x00 | 0x00 | 0x00 |
| Input Data  | S    | ID   | D0   | D1   | D2   | D3   |
| Value       | 0x01 | 0x21 | 0x00 | 0x00 | 0x00 | 0x00 |

#### 2.1.1.8.2.2 Data Receiving

- Specifying the function with ID or reversing TR
- Getting data is complete when TA == TR
- Data available in input bytes D0 ... D3

**Example:**

Getting address offset (GET\_ADDR, ID 0x20)

Original values:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x00     | 0x00      | 0x00      | 0x00      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |

Writing ID (GET\_ADDR, ID 0x20) or reversing TR (Bit 0 in control byte C), data is read:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x20      | 0xXX      | 0xXX      | 0xXX      | 0xXX      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x20      | 0x78      | 0x56      | 0x34      | 0x12      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |

**2.1.1.8.2.3 Writing Data in Safe Configuration Area**

- Writing address offset of password
- Writing password
- Writing address offset of configuration data
- Writing configuration data
- Writing address offset of password
- Resetting password

**Example:**

Writing 0x00000E10 ≈ 3600 (CET) as time zone (address 0x00000308)

Original values:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x00     | 0x00      | 0x00      | 0x00      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x00     | 0x00      | 0x00      | 0x00      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |

Setting address offset of password:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x00     | 0x00      | 0x10      | 0x03      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
| <b>Value</b>       | 0x00     | 0x00      | 0x00      | 0x00      | 0x00      | 0x00      |

Transmitting address offset of password (SET\_ADDR, ID = 0x21):

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x21      | 0x10      | 0x03      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
| <b>Value</b>       | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

Setting password:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x21      | 0xF1      | 0xD0      | 0xA8      | 0x3E      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
| <b>Value</b>       | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

Transmitting password (SET\_CNF, ID = 0x3A):

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x3A      | 0xF1      | 0xD0      | 0xA8      | 0x3E      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
| <b>Value</b>       | 0x01     | 0x3A      | 0x00      | 0x00      | 0x00      | 0x00      |

Setting address offset of time zone:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x3A      | 0x08      | 0x03      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
| <b>Value</b>       | 0x01     | 0x3A      | 0x00      | 0x00      | 0x00      | 0x00      |

Transmitting address offset of time zone (SET\_ADDR, ID = 0x21):

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x21      | 0x08      | 0x03      | 0x00      | 0x00      |
| <b>Input Data</b>  | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
| <b>Value</b>       | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

Setting time zone:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x21      | 0x10      | 0x0E      | 0x00      | 0x00      |

| <b>Input Data</b> | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|-------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>      | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

Transmitting time zone (SET\_CNF, ID = 0x3A):

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x3A      | 0x10      | 0x0E      | 0x00      | 0x00      |

| <b>Input Data</b> | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|-------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>      | 0x01     | 0x3A      | 0x00      | 0x00      | 0x00      | 0x00      |

Setting address offsets of password:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x00     | 0x3A      | 0x10      | 0x03      | 0x00      | 0x00      |

| <b>Input Data</b> | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|-------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>      | 0x00     | 0x3A      | 0x00      | 0x00      | 0x00      | 0x00      |

Transmitting address offsets of password (SET\_ADDR, ID = 0x21):

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x21      | 0x10      | 0x03      | 0x00      | 0x00      |

| <b>Input Data</b> | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|-------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>      | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

Resetting password:

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

| <b>Input Data</b> | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|-------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>      | 0x01     | 0x21      | 0x00      | 0x00      | 0x00      | 0x00      |

Transmitting password (SET\_CNF, ID = 0x3A):

| <b>Output data</b> | <b>C</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|--------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>       | 0x01     | 0x3A      | 0x00      | 0x00      | 0x00      | 0x00      |

| <b>Input Data</b> | <b>S</b> | <b>ID</b> | <b>D0</b> | <b>D1</b> | <b>D2</b> | <b>D3</b> |
|-------------------|----------|-----------|-----------|-----------|-----------|-----------|
| <b>Value</b>      | 0x01     | 0x3A      | 0x00      | 0x00      | 0x00      | 0x00      |

### 2.1.1.9 Connection Examples

#### 2.1.1.9.1 Antenna Supply with 24 V

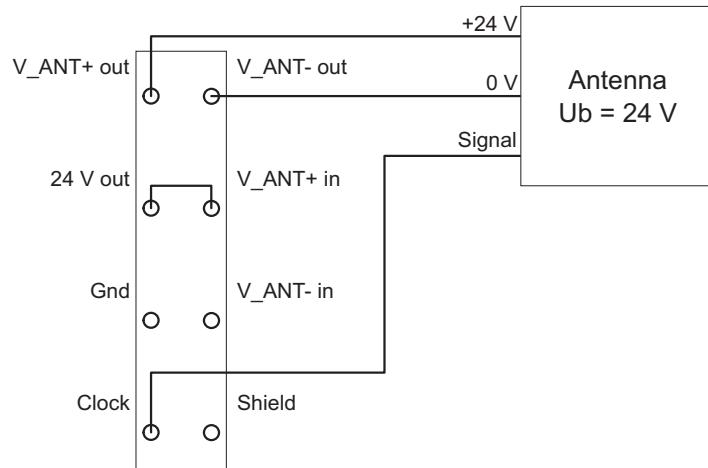


Fig. 2.1.1-5: Antenna Supply with 24 V

g064003e

#### 2.1.1.9.2 Antenna Supply with 3 V via External Power Supply

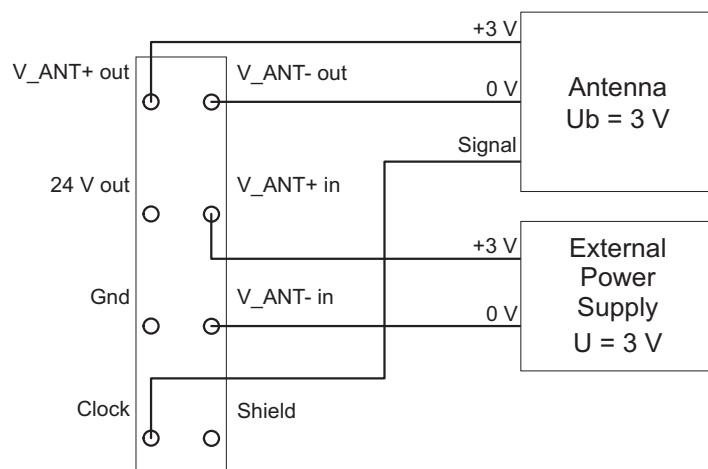


Fig. 2.1.1-6: Antenna Supply via external 3 V

g064004e

## 3 Attachment

### 3.1 Functions

#### 3.1.1 Function Overview

| ID                             | Function           | Description   | Page |
|--------------------------------|--------------------|---|------|
| 0x00                           | IDLE               | No function call  | 33   |
| <b>Date and Time Functions</b> |                    |   |      |
| 0x01                           | GET_TOD100         | Getting time in 100µs since 00:00                                   | 34   |
| 0x09                           | GET_CURRENT_TOD100 | Getting time in 100µs since 00:00, (consecutively)                  | 35   |
| 0x02                           | GET_TOD            | Getting time in milliseconds since 00:00                            | 36   |
| 0x0A                           | GET_CURRENT_TOD    | Getting time in milliseconds since 00:00 (consecutively)            | 37   |
| 0x03                           | GET_DT             | Getting time in seconds since 1.1.1970 00:00                        | 38   |
| 0x0B                           | GET_CURRENT_DT     | Getting time in seconds since 1.1.1970 00:00, (consecutively)       | 39   |
| 0x04                           | GET_UTC            | Getting time in seconds since 1.1.1970 00:00 (UTC)                  | 40   |
| 0x0C                           | GET_CURRENT_UTC    | Getting time in seconds since 1.1.1970 00:00 (UTC), (consecutively) | 41   |
| 0x05                           | GET_DATE           | Getting date in seconds since 1.1.1970 00:00                        | 42   |
| 0x0D                           | GET_CURRENT_DATE   | Getting date in seconds since 1.1.1970 00:00, (consecutively)       | 43   |
| 0x06                           | GET_MS             | Getting millisecond part of the last GET_DT request                 | 44   |
| 0x07                           | GET_WEEKDAY        | Getting weekday   | 45   |
| 0x0F                           | SET_DT             | Setting time in seconds since 1.1.1970 00:00                        | 46   |
| <b>Timer Functions</b>         |                    |   |      |
| 0x10                           | GET_CHANNEL        | Getting switching states of the switching channels                  | 47   |
| 0x11                           | SET_CHANNEL        | Setting switching states of the switching channels                  | 48   |
| 0x12                           | CLEAR_CHANNEL      | Clearing switching states of the switching channels                 | 49   |
| 0x13                           | GET_AUTO           | Getting automatic states of the switching channels                  | 50   |
| 0x14                           | SET_AUTO           | Setting automatic states of the switching channels                  | 51   |

| ID                                  | Function          | Description  | Page |
|-------------------------------------|-------------------|--|------|
| <b>Address Functions</b>            |                   |  |      |
| 0x20                                | GET_ADDR          | Getting the address offset for configuration data or hours-run meter | 52   |
| 0x21                                | SET_ADDR          | Setting address offset for configuration data or service data        | 53   |
| <b>Service Data Functions</b>       |                   |  |      |
| 0x30                                | GET_OPH           | Getting service data   | 54   |
| 0x31                                | GET_OPH_INC       | Getting service data, (auto incremental)                             | 55   |
| 0x32                                | SET_OPH           | Setting service data   | 56   |
| 0x33                                | SET_OPH_INC       | Setting service data, (auto incremental)                             | 57   |
| 0x34                                | GET_SERVICE_FLAGS | Getting service indication of the hours-run meter                    | 58   |
| <b>Configuration Data Functions</b> |                   |  |      |
| 0x38                                | GET_CNF           | Getting configuration data   | 59   |
| 0x39                                | GET_CNF_INC       | Getting configuration data, (auto incremental)                       | 60   |
| 0x3A                                | SET_CNF           | Setting configuration data   | 61   |
| 0x3B                                | SET_CNF_INC       | Setting configuration data, (auto incremental)                       | 62   |
| 0xFF                                | GET_DIAG          | Getting self-test diagnostic   | 63   |

### 3.1.2 Function Overview sorted by ID

| ID   | Function           | Description  | Page |
|------|--------------------|--|------|
| 0x00 | IDLE               | No function call   | 33   |
| 0x01 | GET_TOD100         | Getting time in 100µs since 00:00                                    | 34   |
| 0x02 | GET_TOD            | Getting time in milliseconds since 00:00                             | 36   |
| 0x03 | GET_DT             | Getting time in seconds since 1.1.1970 00:00                         | 38   |
| 0x04 | GET_UTC            | Getting time in seconds since 1.1.1970 00:00 (UTC)                   | 40   |
| 0x05 | GET_DATE           | Getting date in seconds since 1.1.1970 00:00                         | 42   |
| 0x06 | GET_MS             | Getting millisecond part of the last GET_DT request                  | 44   |
| 0x07 | GET_WEEKDAY        | Getting weekday  | 45   |
| 0x09 | GET_CURRENT_TOD100 | Getting time in 100µs since 00:00, (consecutively)                   | 35   |
| 0x0A | GET_CURRENT_TOD    | Getting time in milliseconds since 00:00 (consecutively)             | 37   |
| 0x0B | GET_CURRENT_DT     | Getting time in seconds since 1.1.1970 00:00, (consecutively)        | 39   |
| 0x0C | GET_CURRENT_UTC    | Getting time in seconds since 1.1.1970 00:00 (UTC), (consecutively)  | 41   |
| 0x0D | GET_CURRENT_DATE   | Getting date in seconds since 1.1.1970 00:00, (consecutively)        | 43   |
| 0x0F | SET_DT             | Setting time in seconds since 1.1.1970 00:00                         | 46   |
| 0x10 | GET_CHANNEL        | Getting switching states of the switching channels                   | 47   |
| 0x11 | SET_CHANNEL        | Setting switching states of the switching channels                   | 48   |
| 0x12 | CLEAR_CHANNEL      | Clearing switching states of the switching channels                  | 49   |
| 0x13 | GET_AUTO           | Getting automatic states of the switching channels                   | 50   |
| 0x14 | SET_AUTO           | Setting automatic states of the switching channels                   | 51   |
| 0x20 | GET_ADDR           | Getting the address offset for configuration data or hours-run meter | 52   |
| 0x21 | SET_ADDR           | Setting address offset for configuration data or service data        | 53   |
| 0x30 | GET_OPH            | Getting service data   | 54   |
| 0x31 | GET_OPH_INC        | Getting service data, (auto incremental)                             | 55   |
| 0x32 | SET_OPH            | Setting service data   | 56   |

| <b>ID</b> | <b>Function</b>   | <b>Description</b>                                | <b>Page</b> |
|-----------|-------------------|---|-------------|
| 0x33      | SET_OPH_INC       | Setting service data, (auto incremental)          | 57          |
| 0x34      | GET_SERVICE_FLAGS | Getting service indication of the hours-run meter | 58          |
| 0x38      | GET_CNF           | Getting configuration data                        | 59          |
| 0x39      | GET_CNF_INC       | Getting configuration data, (auto incremental)    | 60          |
| 0x3A      | SET_CNF           | Setting configuration data                        | 61          |
| 0x3B      | SET_CNF_INC       | Setting configuration data, (auto incremental)    | 62          |
| 0xFF      | GET_DIAG          | Getting self-test diagnostic                      | 63          |

### 3.1.3 Function Overview sorted by Function

| ID   | Function           | Description  | Page |
|------|--------------------|--|------|
| 0x12 | CLEAR_CHANNEL      | Clearing switching states of the switching channels                  | 49   |
| 0x20 | GET_ADDR           | Getting the address offset for configuration data or hours-run meter | 52   |
| 0x13 | GET_AUTO           | Getting automatic states of the switching channels                   | 50   |
| 0x10 | GET_CHANNEL        | Getting switching states of the switching channels                   | 47   |
| 0x38 | GET_CNF            | Getting configuration data   | 59   |
| 0x39 | GET_CNF_INC        | Getting configuration data, (auto incremental)                       | 60   |
| 0x0D | GET_CURRENT_DATE   | Getting date in seconds since 1.1.1970 00:00, (consecutively)        | 43   |
| 0x0B | GET_CURRENT_DT     | Getting time in seconds since 1.1.1970 00:00, (consecutively)        | 39   |
| 0x0A | GET_CURRENT_TOD    | Getting time in milliseconds since 00:00 (consecutively)             | 37   |
| 0x09 | GET_CURRENT_TOD100 | Getting time in 100µs since 00:00, (consecutively)                   | 35   |
| 0x0C | GET_CURRENT_UTC    | Getting time in seconds since 1.1.1970 00:00 (UTC), (consecutively)  | 41   |
| 0x05 | GET_DATE           | Getting date in seconds since 1.1.1970 00:00                         | 42   |
| 0xFF | GET_DIAG           | Getting self-test diagnostic   | 63   |
| 0x03 | GET_DT             | Getting time in seconds since 1.1.1970 00:00                         | 38   |
| 0x06 | GET_MS             | Getting millisecond part of the last GET_DT request                  | 44   |
| 0x30 | GET_OPH            | Getting service data   | 54   |
| 0x31 | GET_OPH_INC        | Getting service data, (auto incremental)                             | 55   |
| 0x34 | GET_SERVICE_FLAGS  | Getting service indication of the hours-run meter                    | 58   |
| 0x02 | GET_TOD            | Getting time in milliseconds since 00:00                             | 36   |
| 0x01 | GET_TOD100         | Getting time in 100µs since 00:00                                    | 34   |
| 0x04 | GET_UTC            | Getting time in seconds since 1.1.1970 00:00 (UTC)                   | 40   |
| 0x07 | GET_WEEKDAY        | Getting weekday  | 45   |
| 0x00 | IDLE               | No function call   | 33   |
| 0x21 | SET_ADDR           | Setting address offset for configuration data or service data        | 53   |

| <b>ID</b> | <b>Function</b> | <b>Description</b>                                 | <b>Page</b> |
|-----------|-----------------|--|-------------|
| 0x14      | SET_AUTO        | Setting automatic states of the switching channels | 51          |
| 0x11      | SET_CHANNEL     | Setting switching states of the switching channels | 48          |
| 0x3A      | SET_CNF         | Setting configuration data                         | 61          |
| 0x3B      | SET_CNF_INC     | Setting configuration data, (auto incremental)     | 62          |
| 0x0F      | SET_DT          | Setting time in seconds since 1.1.1970 00:00       | 46          |
| 0x32      | SET_OPH         | Setting service data                               | 56          |
| 0x33      | SET_OPH_INC     | Setting service data, (auto incremental)           | 57          |

### 3.1.4 Function Reference

#### 3.1.4.1 IDLE, 0x00

No request will be executed using the IDLE function.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x00</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any original value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x00</b> |       |       |       |       |       |       |       |
| D0                | 0xXX        |       |       |       |       |       |       |       |
| D1                | 0xXX        |       |       |       |       |       |       |       |
| D2                | 0xXX        |       |       |       |       |       |       |       |
| D3                | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

#### Example:

The function can be called, for example, to reverse the toggle flag without effects or to stop the process of consecutive functions.

### 3.1.4.2 GET\_TOD100, 0x01

The current time is read using the GET\_TOD100 function.  
The input data contains the current time in 100μs steps.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x01</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                  |       |       |       |       |       |       |       |
|-------------------|------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$            | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x01</b>      |       |       |       |       |       |       |       |
| D0                | Time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...              |       |       |       |       |       |       |       |
| D2                | ...              |       |       |       |       |       |       |       |
| D3                | Time (high byte) |       |       |       |       |       |       |       |

#### Example:

$$08:45:35 \doteq ((8*60+45)*60+35)*10000 = 315350000 \doteq 0x12CBDBF0$$

Input data: D0 = 0xF0, D1 = 0xDB, D2 = 0xCB, D3 = 0x12

**3.1.4.3 GET\_CURRENT\_TOD100, 0x09**

The current time is read using the GET\_CURRENT\_TOD100 function. The value is updated in each bus cycle without requiring a new function call. The input data contains the current time in 100µs steps.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x09</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                  |       |       |       |       |       |       |       |
|-------------------|------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$            | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x09</b>      |       |       |       |       |       |       |       |
| D0                | Time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...              |       |       |       |       |       |       |       |
| D2                | ...              |       |       |       |       |       |       |       |
| D3                | Time (high byte) |       |       |       |       |       |       |       |

**Example:**

$$08:45:35 \triangleq ((8*60+45)*60+35)*10000 = 315350000 \triangleq 0x12CBDBF0$$

Input data: D0 = 0xF0, D1 = 0xDB, D2 = 0xCB, D3 = 0x12

### 3.1.4.4 GET\_TOD, 0x02

The current time is read using the GET\_TOD function.  
The input data contains the current time in 1ms steps.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x02</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                  |       |       |       |       |       |       |       |
|-------------------|------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$            | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x02</b>      |       |       |       |       |       |       |       |
| D0                | Time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...              |       |       |       |       |       |       |       |
| D2                | ...              |       |       |       |       |       |       |       |
| D3                | Time (high byte) |       |       |       |       |       |       |       |

#### Example:

$$08:45:35 \doteq ((8*60+45)*60+35)*1000 = 31535000 \doteq 0x01E12F98$$

Input data: D0 = 0x98, D1 = 0x2F, D2 = 0xE1, D3 = 0x01

**3.1.4.5 GET\_CURRENT\_TOD, 0x0A**

The GET\_CURRENT\_TOD function is used to read the current time. The value is updated in each bus cycle without requiring a new function call. The input data contains the current time in 1ms steps.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x0A</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                  |       |       |       |       |       |       |       |
|-------------------|------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$            | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x0A</b>      |       |       |       |       |       |       |       |
| D0                | Time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...              |       |       |       |       |       |       |       |
| D2                | ...              |       |       |       |       |       |       |       |
| D3                | Time (high byte) |       |       |       |       |       |       |       |

**Example:**

$$08:45:35 \doteq ((8*60+45)*60+35)*1000 = 31535000 \doteq 0x01E12F98$$

Input data: D0 = 0x98, D1 = 0x2F, D2 = 0xE1, D3 = 0x01

### 3.1.4.6 GET\_DT, 0x03

Both the current date and time are read using the GET\_DT function.  
The input data contains the seconds that have elapsed since 01.01.1970 00:00 at local time.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x03</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x03</b>               |       |       |       |       |       |       |       |
| D0                | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                       |       |       |       |       |       |       |       |
| D2                | ...                       |       |       |       |       |       |       |       |
| D3                | Date and time (high byte) |       |       |       |       |       |       |       |

#### Example:

27.03.2003 16:06:23  $\hat{=}$  1048781183  $\hat{=}$  0x3E83217F

Input data: D0 = 0x7F, D1 = 0x21, D2 = 0x83, D3 = 0x3E

**3.1.4.7 GET\_CURRENT\_DT, 0x0B**

Both the current date and time are read using the GET\_CURRENT\_DT function.

The value is updated in each bus cycle without requiring a new function call. The input data contains the seconds that have elapsed since 01.01.1970 00:00 in local time.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x0B</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x0B</b>               |       |       |       |       |       |       |       |
| D0                | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                       |       |       |       |       |       |       |       |
| D2                | ...                       |       |       |       |       |       |       |       |
| D3                | Date and time (high byte) |       |       |       |       |       |       |       |

**Example:**

27.03.2003 16:06:23 ≈ 1048781183 ≈ 0x3E83217F

Input data: D0 = 0x7F, D1 = 0x21, D2 = 0x83, D3 = 0x3E

### 3.1.4.8 GET\_UTC, 0x04

Both the current date and time are read using the GET\_UTC function.  
The input data contains the seconds that have elapsed since 01.01.1970 00:00 in UTC (Coordinated Universal Time, similar to Greenwich Mean Time).

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x04</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x04</b>               |       |       |       |       |       |       |       |
| D0                | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                       |       |       |       |       |       |       |       |
| D2                | ...                       |       |       |       |       |       |       |       |
| D3                | Date and time (high byte) |       |       |       |       |       |       |       |

#### Example:

Summer time in Germany (MESZ)

27.03.2003 16:06:23  $\doteq$   $1048781183 - 2 * 3600 = 1048773983 \doteq 0x3E83055F$

Input data: D0 = 0x5F, D1 = 0x05, D2 = 0x83, D3 = 0x3E

### 3.1.4.9 GET\_CURRENT\_UTC, 0x0C

Both the current date and time are read using the GET\_CURRENT\_UTC function.

The value is updated in each bus cycle without requiring a new function call. The input data contains the seconds that have elapsed since 01.01.1970 00:00 in UTC (Coordinated Universal Time, similar to Greenwich Mean Time).

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x0C</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x0C</b>               |       |       |       |       |       |       |       |
| D0                | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                       |       |       |       |       |       |       |       |
| D2                | ...                       |       |       |       |       |       |       |       |
| D3                | Date and time (high byte) |       |       |       |       |       |       |       |

#### Example:

Summer time in Germany (MESZ)

27.03.2003 16:06:23  $\doteq$  1048781183 – 2\*3600 = 1048773983  $\doteq$  0x3E83055F

Input data: D0 = 0x5F, D1 = 0x05, D2 = 0x83, D3 = 0x3E

### 3.1.4.10 GET\_DATE, 0x05

The current date is read using the GET\_DATE function.  
The input data contains the seconds that have elapsed from 01.01.1970 00:00 to 00:00 of the current day in local time.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x05</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x05</b>               |       |       |       |       |       |       |       |
| D0                | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                       |       |       |       |       |       |       |       |
| D2                | ...                       |       |       |       |       |       |       |       |
| D3                | Date and time (high byte) |       |       |       |       |       |       |       |

#### Example:

27.03.2003 16:06:23 → 27.03.2003 00:00:00 ≈ 1048723200 ≈ 0x3E823F00

Input data: D0 = 0x00, D1 = 0x3F, D2 = 0x82, D3 = 0x3E

**3.1.4.11 GET\_CURRENT\_DATE, 0x0D**

The GET\_CURRENT\_DATE function is used to get the current date. The value is updated in each bus cycle without requiring a new function call. The input data contains the seconds that have elapsed from 01.01.1970 00:00 to 00:00 of the current day in local time.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x0D</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                           |       |       |       |       |       |       |       |
|-------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x0D</b>               |       |       |       |       |       |       |       |
| D0                | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                       |       |       |       |       |       |       |       |
| D2                | ...                       |       |       |       |       |       |       |       |
| D3                | Date and time (high byte) |       |       |       |       |       |       |       |

**Example:**

27.03.2003 16:06:23 → 27.03.2003 00:00:00 ≈ 1048723200 ≈ 0x3E823F00

Input data: D0 = 0x00, D1 = 0x3F, D2 = 0x82, D3 = 0x3E

### 3.1.4.12 GET\_MS, 0x06

The GET\_MS function is used to get the millisecond part of the last GET\_DT call.

The millisecond part is stored in the input bytes D0 and D1. The input bytes D2 and D3 are reserved.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x06</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                          |       |       |       |       |       |       |       |
|-------------------|--------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                    | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                        | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x06</b>              |       |       |       |       |       |       |       |
| D0                | Milliseconds (low byte)  |       |       |       |       |       |       |       |
| D1                | Milliseconds (high byte) |       |       |       |       |       |       |       |
| D2                | 0x00                     |       |       |       |       |       |       |       |
| D3                | 0x00                     |       |       |       |       |       |       |       |

#### Example:

27.03.2003 16:06:23.462 → 462 ≈ 0x0000001CE

Input data: D0 = 0xCE, D1 = 0x01, D2 = 0x00, D3 = 0x00

### 3.1.4.13 GET\_WEEKDAY, 0x07

The GET\_WEEKDAY function is used to get the current weekday. The number of the weekday is stored in the input byte D0. The inputs bytes D1 ... D3 are reserved.

(0 ≈ Monday, 1 ≈ Tuesday, ... , 5 ≈ Saturday, 6 ≈ Sunday)

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x07</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                 |       |       |       |       |       |       |       |
|-------------------|-----------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$           | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0               | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x07</b>     |       |       |       |       |       |       |       |
| D0                | Day of the week |       |       |       |       |       |       |       |
| D1                | 0x00            |       |       |       |       |       |       |       |
| D2                | 0x00            |       |       |       |       |       |       |       |
| D3                | 0x00            |       |       |       |       |       |       |       |

#### Example:

Wednesday ≈ 2

Input data: D0 = 0x02, D1 = 0x00, D2 = 0x00, D3 = 0x00

### 3.1.4.14 SET\_DT, 0x0F

The current date and the time are set using the SET\_DT function.  
The local time that has elapsed in seconds from 01.01.1970 00:00 is indicated in the output data.

| Function Request |                           |       |       |       |       |       |       |       |
|------------------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$                     | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0                         | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x0F</b>               |       |       |       |       |       |       |       |
| D0               | Date and time (low byte)  |       |       |       |       |       |       |       |
| D1               | ...                       |       |       |       |       |       |       |       |
| D2               | ...                       |       |       |       |       |       |       |       |
| D3               | Date and time (high byte) |       |       |       |       |       |       |       |

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x0F</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

#### Example:

27.03.2003 16:06:23 ≈ 1048781183 ≈ 0x3E83217F

Output data: D0 = 0x7F, D1 = 0x21, D2 = 0x83, D3 = 0x3E

**3.1.4.15 GET\_CHANNEL, 0x10**

The GET\_CHANNEL function is used to get the switching status of the 32 switching channels.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x10</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x10</b> |       |       |       |       |       |       |       |
| D0                | C7          | C6    | C5    | C4    | C3    | C2    | C1    | C0    |
| D1                | C15         | C14   | C13   | C12   | C11   | C10   | C9    | C8    |
| D2                | C23         | C22   | C21   | C20   | C19   | C18   | C17   | C16   |
| D3                | C31         | C30   | C29   | C28   | C27   | C26   | C25   | C24   |

Cx Switching channel 0 – switching channel 31

**Example:**

The switching channels 1, 17, 19, 20, 21, 22, 23 are on, all other switching channels are off  $\cong$  (0x00FA0002):

Input data: D0 = 0x02, D1 = 0x00, D2 = 0xFA, D3 = 0x00

### 3.1.4.16 SET\_CHANNEL, 0x11

Using the SET\_CHANNEL function, the switching states of the switching channels that are described by the set bits are set to one.

In automatic mode, the switching states will be overwritten at the next alarm event if required.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x11</b> |       |       |       |       |       |       |       |
| D0               | C7          | C6    | C5    | C4    | C3    | C2    | C1    | C0    |
| D1               | C15         | C14   | C13   | C12   | C11   | C10   | C9    | C8    |
| D2               | C23         | C22   | C21   | C20   | C19   | C18   | C17   | C16   |
| D3               | C31         | C30   | C29   | C28   | C27   | C26   | C25   | C24   |

Cx      Switching channel 0 – switching channel 31

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x11</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

#### Example:

Switching channels 8, 10, 12 and 14 must be set  $\triangleq$  (0x00005500):

Output data: D0 = 0x00, D1 = 0x55, D2 = 0x00, D3 = 0x00

### 3.1.4.17 CLEAR\_CHANNEL, 0x12

Using the CLEAR\_CHANNEL function, the switching states of the switching channels that are described by the set bits are set to zero.

In automatic mode, the switching states will be overwritten at the next alarm event if necessary.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x12</b> |       |       |       |       |       |       |       |
| D0               | C7          | C6    | C5    | C4    | C3    | C2    | C1    | C0    |
| D1               | C15         | C14   | C13   | C12   | C11   | C10   | C9    | C8    |
| D2               | C23         | C22   | C21   | C20   | C19   | C18   | C17   | C16   |
| D3               | C31         | C30   | C29   | C28   | C27   | C26   | C25   | C24   |

Cx      Switching channel 0 – switching channel 31

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x12</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

#### Example:

Switching channels 17, 19, 21 and 23 must be set  $\cong$  (0x00AA0000):

Output data: D0 = 0x00, D1 = 0x00, D2 = 0xAA, D3 = 0x00

### 3.1.4.18 GET\_AUTO, 0x13

The GET\_AUTO function is used to get the automatic status of the 32 switching channels. The automatic mode is switched off for the switching channels that are described by the set bits.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x13</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x13</b> |       |       |       |       |       |       |       |
| D0                | C7          | C6    | C5    | C4    | C3    | C2    | C1    | C0    |
| D1                | C15         | C14   | C13   | C12   | C11   | C10   | C9    | C8    |
| D2                | C23         | C22   | C21   | C20   | C19   | C18   | C17   | C16   |
| D3                | C31         | C30   | C29   | C28   | C27   | C26   | C25   | C24   |

Cx Switching channel 0 – switching channel 31

#### Example:

The switching channels 2, 4 and 6 are in manual mode, all other switching channels are in automatic mode  $\hat{=}$  (0x00000054):

Input data: D0 = 0x54, D1 = 0x00, D2 = 0x00, D3 = 0x00

**3.1.4.19 SET\_AUTO, 0x14**

Using the SET\_AUTO function, the automatic mode is switched off for the switching channels that are described by the set bits.

The automatic mode is activated for all other switching channels.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x14</b> |       |       |       |       |       |       |       |
| D0               | C7          | C6    | C5    | C4    | C3    | C2    | C1    | C0    |
| D1               | C15         | C14   | C13   | C12   | C11   | C10   | C9    | C8    |
| D2               | C23         | C22   | C21   | C20   | C19   | C18   | C17   | C16   |
| D3               | C31         | C30   | C29   | C28   | C27   | C26   | C25   | C24   |

Cx      Switching channel 0 – switching channel 31

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x14</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

**Example:**

For the switching channels 9, 11 and 13 the automatic mode must be switched off  $\triangleq$  (0x00002A00):

Output data: D0 = 0x00, D1 = 0x2A, D2 = 0x00, D3 = 0x00

### 3.1.4.20 GET\_ADDR, 0x20

The GET\_ADDR function is used to get the address offset for a referenced function call, for example, when accessing the configuration data or service data (hours-run meter and maintenance interval).

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x20</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                            |       |       |       |       |       |       |       |
|-------------------|----------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                      | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                          | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x20</b>                |       |       |       |       |       |       |       |
| D0                | Address offset (low byte)  |       |       |       |       |       |       |       |
| D1                | ...                        |       |       |       |       |       |       |       |
| D2                | ...                        |       |       |       |       |       |       |       |
| D3                | Address offset (high byte) |       |       |       |       |       |       |       |

#### Example:

Address offset contains the value 0x12345678:

Input data: D0 = 0x78, D1 = 0x56, D2 = 0x34, D3 = 0x12

### 3.1.4.21 SET\_ADDR, 0x21

Using the SET\_ADDR function, the address offset for a referenced function call is written, for example, when accessing the configuration data or the service data (hours-run meter and maintenance interval).

| Function Request |                            |       |       |       |       |       |       |
|------------------|----------------------------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$                      | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ |
| C                | 0                          | 0     | 0     | 0     | 0     | 0     | 0     |
| ID               | <b>0x21</b>                |       |       |       |       |       |       |
| D0               | Address offset (low byte)  |       |       |       |       |       |       |
| D1               | ...                        |       |       |       |       |       |       |
| D2               | ...                        |       |       |       |       |       |       |
| D3               | Address offset (high byte) |       |       |       |       |       |       |

| Function Response |             |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     |
| ID                | <b>0x21</b> |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |

#### Example:

Reading address offset 0x12345678:

Output data: D0 = 0x78, D1 = 0x56, D2 = 0x34, D3 = 0x12

### 3.1.4.22 GET\_OPH, 0x30

The GET\_OPH function is used to get the service data of the 32 switching channels (hours-run meter and maintenance interval).

The design of the service data is described in section 2.1.1.6.3, „Service Dat“. Before calling this function for the first time, the required address offset must be set for the hours run or maintenance interval of the appropriate meter using the SET\_ADDR function.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x30</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x30</b> |       |       |       |       |       |       |       |
| D0                | (Low byte)  |       |       |       |       |       |       |       |
| D1                | ...         |       |       |       |       |       |       |       |
| D2                | ...         |       |       |       |       |       |       |       |
| D3                | (High byte) |       |       |       |       |       |       |       |

**3.1.4.23 GET\_OPH\_INC (auto incremental), 0x31**

The GET\_OPH\_INC function is used to get the service data of the 32 switching channels (hours-run meter and maintenance interval).

The design of the service data is described in section 2.1.1.6.3, „Service Dat“. Before calling this function for the first time, the required address offset must be set for the hours run or maintenance interval of the appropriate meter using the SET\_ADDR function.

When a new function call is made, 4 bytes are automatically added to the address offset.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x31</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x31</b> |       |       |       |       |       |       |       |
| D0                | (Low byte)  |       |       |       |       |       |       |       |
| D1                | ...         |       |       |       |       |       |       |       |
| D2                | ...         |       |       |       |       |       |       |       |
| D3                | (High byte) |       |       |       |       |       |       |       |

### 3.1.4.24 SET\_OPH, 0x32

The SET\_OPH function is used to get the service data of the 32 switching channels (hours-run meter and maintenance interval).

The design of the service data is described in section 2.1.1.6.3, „Service Dat“. Before calling this function for the first time, the required address offset must be set for the hours run or maintenance interval of the appropriate meter using the SET\_ADDR function.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x32</b> |       |       |       |       |       |       |       |
| D0               | (Low byte)  |       |       |       |       |       |       |       |
| D1               | ...         |       |       |       |       |       |       |       |
| D2               | ...         |       |       |       |       |       |       |       |
| D3               | (High byte) |       |       |       |       |       |       |       |

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x32</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

### 3.1.4.25 SET\_OPH\_INC (auto incremental), 0x33

The SET\_OPH\_INC function is used to get the service data of the 32 switching channels (hours-run meter and maintenance interval).

The design of the service data is described in section 2.1.1.6.3, „Service Dat“. Before calling this function for the first time, the required address offset must be set for the hours run or maintenance interval of the appropriate meter using the SET\_ADDR function.

When a new function call is made, 4 bytes are automatically added to the address offset.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | 0x33        |       |       |       |       |       |       |       |
| D0               | (Low byte)  |       |       |       |       |       |       |       |
| D1               | ...         |       |       |       |       |       |       |       |
| D2               | ...         |       |       |       |       |       |       |       |
| D3               | (High byte) |       |       |       |       |       |       |       |

| Function Response |       |       |       |       |       |       |       |       |
|-------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0     | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | 0x33  |       |       |       |       |       |       |       |
| D0                | 0x00  |       |       |       |       |       |       |       |
| D1                | 0x00  |       |       |       |       |       |       |       |
| D2                | 0x00  |       |       |       |       |       |       |       |
| D3                | 0x00  |       |       |       |       |       |       |       |

### 3.1.4.26 GET\_SERVICE\_FLAGS, 0x34

The GET\_SERVICE\_FLAGS function is used to get the status of the maintenance indicators of the 32 switching channels.

The bits that are set in the input data correspond to the switching channels whose set maintenance interval is exceeded.

To delete a service flag, the relevant hours-run meter must be set with a value that is smaller than the corresponding maintenance interval.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x34</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x34</b> |       |       |       |       |       |       |       |
| D0                | C7          | C6    | C5    | C4    | C3    | C2    | C1    | C0    |
| D1                | C15         | C14   | C13   | C12   | C11   | C10   | C9    | C8    |
| D2                | C23         | C22   | C21   | C20   | C19   | C18   | C17   | C16   |
| D3                | C31         | C30   | C29   | C28   | C27   | C26   | C25   | C24   |

Cx Switching channel 0 – switching channel 31

#### Example:

Maintenance interval switching channel 1, 17, 19, 20, 21, 22, 23 is exceeded  
 $\triangleq$  (0x00FA0002):

Input data: D0 = 0x02, D1 = 0x00, D2 = 0xFA, D3 = 0x00

Deleting requires that the relevant meters have been set to 0.

### 3.1.4.27 GET\_CNF, 0x38

The GET\_CNF function is used to get the configuration data.

The design of the configuration data is described in section 2.1.1.6.2, „Configuration Dat“.

Before calling this function for the first time, the required address offset must be set for configuration data of the appropriate meter using the SET\_ADDR function.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x38</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x38</b> |       |       |       |       |       |       |       |
| D0                | (Low byte)  |       |       |       |       |       |       |       |
| D1                | ...         |       |       |       |       |       |       |       |
| D2                | ...         |       |       |       |       |       |       |       |
| D3                | (High byte) |       |       |       |       |       |       |       |

### 3.1.4.28 GET\_CNF\_INC (auto incremental), 0x39

The GET\_CNF\_INC function is used to get the configuration data. The design of the configuration data is described in section 2.1.1.6.2, „Configuration Dat“.  
Before calling this function for the first time, the required address offset must be set for configuration data of the appropriate meter using the SET\_ADDR function.  
When a new function call is made, 4 bytes are automatically added to the address offset.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x39</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x39</b> |       |       |       |       |       |       |       |
| D0                | (Low byte)  |       |       |       |       |       |       |       |
| D1                | ...         |       |       |       |       |       |       |       |
| D2                | ...         |       |       |       |       |       |       |       |
| D3                | (High byte) |       |       |       |       |       |       |       |

### 3.1.4.29 SET\_CNF, 0x3A

The SET\_CNF function is used to get the configuration data.

The design of the configuration data is described in section 2.1.1.6.2, „Configuration Dat“.

Before calling this function for the first time, the required address offset must be set for configuration data of the appropriate meter using the SET\_ADDR function.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x3A</b> |       |       |       |       |       |       |       |
| D0               | (Low byte)  |       |       |       |       |       |       |       |
| D1               | ...         |       |       |       |       |       |       |       |
| D2               | ...         |       |       |       |       |       |       |       |
| D3               | (High byte) |       |       |       |       |       |       |       |

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x3A</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

### 3.1.4.30 SET\_CNF\_INC (auto incremental), 0x3B

The SET\_CNF\_INC function is used to get the configuration data.

The design of the configuration data is described in section 2.1.1.6.2,  
„Configuration Dat“.

Before calling this function for the first time, the required address offset must be set for configuration data of the appropriate meter using the SET\_ADDR function.

When a new function call is made, 4 bytes are automatically added to the address offset.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0x3B</b> |       |       |       |       |       |       |       |
| D0               | (Low byte)  |       |       |       |       |       |       |       |
| D1               | ...         |       |       |       |       |       |       |       |
| D2               | ...         |       |       |       |       |       |       |       |
| D3               | (High byte) |       |       |       |       |       |       |       |

| Function Response |             |       |       |       |       |       |       |       |
|-------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0x3B</b> |       |       |       |       |       |       |       |
| D0                | 0x00        |       |       |       |       |       |       |       |
| D1                | 0x00        |       |       |       |       |       |       |       |
| D2                | 0x00        |       |       |       |       |       |       |       |
| D3                | 0x00        |       |       |       |       |       |       |       |

**3.1.4.31 GET\_DIAG, 0xFF**

The function GET\_DIAG is used to get the diagnostic status of the module. The diagnostic information is stored in input byte D0. The input bytes D1 ... D3 are reserved.

| Function Request |             |       |       |       |       |       |       |       |
|------------------|-------------|-------|-------|-------|-------|-------|-------|-------|
| Byte             | $2^7$       | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| C                | 0           | 0     | 0     | 0     | 0     | 0     | 0     | TR    |
| ID               | <b>0xFF</b> |       |       |       |       |       |       |       |
| D0               | 0xXX        |       |       |       |       |       |       |       |
| D1               | 0xXX        |       |       |       |       |       |       |       |
| D2               | 0xXX        |       |       |       |       |       |       |       |
| D3               | 0xXX        |       |       |       |       |       |       |       |

0xXX any return value

| Function Response |                         |       |       |       |       |       |       |       |
|-------------------|-------------------------|-------|-------|-------|-------|-------|-------|-------|
| Byte              | $2^7$                   | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| S                 | 0                       | 0     | 0     | 0     | 0     | 0     | 0     | TA    |
| ID                | <b>0xFF</b>             |       |       |       |       |       |       |       |
| D0                | Diagnostics information |       |       |       |       |       |       |       |
| D1                | 0x00                    |       |       |       |       |       |       |       |
| D2                | 0x00                    |       |       |       |       |       |       |       |
| D3                | 0x00                    |       |       |       |       |       |       |       |

The following table shows the signification of the possible diagnostics information.

| Diagnostics Information (Input Byte D0) |  |
|---|--|
| Bit                                     | Diagnostics                              |
| 0                                       | Antenna signal level                     |
| 1                                       | Buffer voltage too small or clock failed |
| 4                                       | Antenna signal not available or defect   |
| 5                                       | Memory is not addressable                |



WAGO Kontakttechnik GmbH & Co. KG  
Postfach 2880 • D-32385 Minden  
Hansastraße 27 • D-32423 Minden  
Phone: 05 71/8 87 – 0  
Fax: 05 71/8 87 – 1 69  
E-Mail: [info@wago.com](mailto:info@wago.com)

Internet: <http://www.wago.com>