



# Transmission protocol and register

## Hardware-Description

2023 March

# INDEX

|  |           |
|--|-----------|
| <b><u>1. Introduction</u></b>                                | <b>7</b>  |
| 1.1. What are registers ?                                    | 8         |
| 1.2. Accessing registers with 8, 16, 32 or 64 bit data width | 8         |
| <b><u>2. Register assignment</u></b>                         | <b>11</b> |
| 2.1. General   | 12        |
| 2.1.1. Reading the number of channels                        | 12        |
| 2.2. Access to in-/outputs                                   | 13        |
| 2.2.1. Register overview                                     | 13        |
| 2.2.2. Register for digital inputs                           | 15        |
| 2.2.2.1. Reading the inputs                                  | 15        |
| 2.2.2.2. Reading the input state changes                     | 16        |
| 2.2.2.3. Reading the input counter                           | 18        |
| 2.2.2.4. Reading and resetting the input counter             | 19        |
| 2.2.2.5. Mode Counter-Modul (only for O8-R8)                 | 21        |
| 2.2.2.6. Register for digital input filter                   | 22        |
| 2.2.3. Register for digital outputs                          | 23        |
| 2.2.3.1. Setting and resetting outputs                       | 23        |
| 2.2.3.2. Readback outputs                                    | 25        |
| 2.2.3.3. Setting outputs                                     | 26        |
| 2.2.3.4. Resetting outputs                                   | 28        |
| 2.2.4. Register for digital PWM outputs (RO-PWM)             | 30        |
| 2.2.4.1. Example for setting a digital PWM output            | 32        |
| 2.2.5. Register for 48 bit input counter                     | 34        |
| 2.2.5.1. 48 bit input counter address range                  | 34        |
| 2.2.5.1.1. 48 bit input counter register                     | 35        |
| 2.2.5.1.1.1, Counter mode                                    | 36        |
| 2.2.5.1.1.1, Sub mode for counter mode COUNT_RISING_EDGE     | 36        |
| 2.2.5.1.1.2, Input filter                                    | 37        |
| 2.2.5.1.2. 48 bit input counter latch register               | 38        |
| 2.2.6. Register for pulse generator outputs (RO-CNT8)        | 39        |
| 2.2.6.1. Pulse generator outputs address range               | 39        |

# INDEX

|  |           |
|--|-----------|
| 2.2.6.1.1. Pulse generator output register                     | 40        |
| 2.2.6.1.2. Example for setting a pulse generator output        | 41        |
| 2.2.7. Register to set the direction of the in-/outputs        | 45        |
| 2.2.7.1. Setting byte by byte the direction of the in-/outputs | 46        |
| 2.2.7.2. Setting bit by bit the direction of the in-/outputs   | 48        |
| 2.2.8. Register for analog inputs (A/D)                        | 50        |
| 2.2.8.1. A/D converter address range                           | 50        |
| 2.2.8.2. A/D converter register                                | 52        |
| 2.2.8.2.1. A/D - mode  | 52        |
| 2.2.9. Register for temperature inputs                         | 53        |
| 2.2.9.1. Temperature input address range                       | 53        |
| 2.2.9.1.1. Temperature input register                          | 55        |
| 2.2.9.1.2. Example for getting temperatures                    | 56        |
| 2.2.10. Register for analog outputs (D/A)                      | 58        |
| 2.2.10.1. D/A converter address range                          | 58        |
| 2.2.10.2. D/A converter register                               | 60        |
| 2.2.10.2.1. D/A - mode   | 61        |
| 2.2.11. Register for stepper                                   | 62        |
| 2.2.11.1. Stepper address range                                | 62        |
| 2.2.11.2. Stepper command register                             | 63        |
| 2.2.11.2.1. Command list                                       | 64        |
| 2.2.11.2.2. Explanation of par 1, 2 and 3                      | 66        |
| 2.2.11.3. Reading stepper status                               | 66        |
| 2.2.11.3.1. Explanation of command GET_POSITION                | 66        |
| 2.2.11.3.2. Explanation of the bits for command GET_ACTIVITY   | 67        |
| 2.2.11.3.3. Explanation of the bits for command GET_SWITCH     | 68        |
| 2.2.12. Timeout-register for digital and analog outputs (D/A)  | 69        |
| 2.2.12.1. Timeout-register                                     | 69        |
| 2.2.12.1.1. Timeout address range for timeout-value            | 69        |
| <b>3. CAN protocol</b>   | <b>70</b> |
| 3.1. Speed and interface parameters                            | 71        |
| 3.2. Structure of the "sending packet"                         | 71        |

# INDEX

|  |            |
|--|------------|
| 3.2.1. COMMAND Write   | 72         |
| 3.2.2. COMMAND Read  | 73         |
| 3.2.3. ADDRESS   | 74         |
| 3.2.4. JOB-ID  | 74         |
| 3.2.5. DATA  | 74         |
| 3.3. Example for a write command                                 | 74         |
| 3.4. Example for writing the first 16 relays                     | 76         |
| 3.5. Structure of the "response packet"                          | 77         |
| 3.5.1. "OK" - response packet                                    | 77         |
| 3.5.2. "Error" - response packet                                 | 78         |
| 3.5.2.1. Error code "0xfd" – JOB-ID double                       | 78         |
| 3.5.2.2. Error code "0xfe" – invalid command                     | 79         |
| 3.6. Structure of the CAN TX/RX packages                         | 80         |
| 3.6.1. Digital inputs  | 80         |
| 3.6.2. Digital outputs   | 81         |
| 3.6.3. Digital input counter (16-bit)                            | 82         |
| 3.6.4. Digital input counter (48-Bit) - 32-Bit package           | 83         |
| 3.6.5. Digital input counter (48-Bit) - 64-Bit package           | 84         |
| 3.6.6. Analog in-/outputs  | 85         |
| 3.6.6.1. Analog inputs   | 85         |
| 3.6.6.2. Analog outputs  | 87         |
| 3.6.6.3. Examples  | 88         |
| 3.6.7. Temperature inputs  | 91         |
| 3.6.8. Stepper   | 94         |
| 3.6.8.1. Command-List  | 95         |
| 3.6.8.2. Values for par 1 of function<br>SET_MOTORCHARACTERISTIC | 97         |
| 3.6.8.3. Values for par 1 of function GO_REFSWITCH               | 99         |
| 3.6.8.4. Examples  | 100        |
| 3.6.9. PWM outputs   | 101        |
| <b>4. Ethernet (TCP) protocol</b>                                | <b>102</b> |

# INDEX

|   |            |
|---|------------|
| 4.1. The TCP protocol                   | 103        |
| 4.1.1. IP-adress                        | 103        |
| 4.1.2. Netmask                          | 103        |
| 4.1.3. Port                             | 103        |
| 4.2. TCP request/response string        | 104        |
| 4.2.1. Request string structure         | 104        |
| 4.2.1.1. JOB-ID                         | 107        |
| 4.2.1.2. Length of the request string   | 107        |
| 4.2.1.3. Extended address mode          | 107        |
| 4.2.1.4. COMMAND                        | 107        |
| 4.2.1.5. WIDTH                          | 107        |
| 4.2.1.6. ADDRESS                        | 109        |
| 4.2.1.7. DATA                           | 109        |
| 4.2.2. Response string structure        | 110        |
| 4.2.2.1. Error code                     | 110        |
| 4.2.2.2. JOB-ID                         | 111        |
| 4.2.2.3. Length of the response string  | 111        |
| 4.2.2.4. DATA                           | 111        |
| <b>5. Serial protocol</b>               | <b>112</b> |
| 5.1. Speed and interface parameters     | 113        |
| 5.2. Structure of the „sending string“  | 113        |
| 5.2.1. Start character                  | 114        |
| 5.2.2. MODUL-NO                         | 115        |
| 5.2.3. JOB-ID                           | 115        |
| 5.2.4. COMMAND                          | 115        |
| 5.2.5. WIDTH                            | 116        |
| 5.2.6. ADDRESS                          | 116        |
| 5.2.7. DATA                             | 116        |
| 5.2.8. CHECKSUM                         | 117        |
| 5.2.9. END                              | 117        |
| 5.3. Structure of the „response string“ | 118        |

# INDEX

|  |     |
|--|-----|
| 5.3.1. Response-string „O“ (OK)                  | 118 |
| 5.3.2. Response-string „D“ (DATA)                | 120 |
| 5.3.3. Start-character with response „E“ (ERROR) | 122 |

## **6. Appendix** **123**

|                                |     |
|--------------------------------|-----|
| 6.1. Revisions                 | 124 |
| 6.2. Copyrights and trademarks | 125 |

# Introduction

---



# **1. Introduction**

## **1.1. What are registers ?**

Registers are little scratch-pad like storages, buffering writing or reading data. The data stays stored until they are overwritten or until their power supply is cut off. They have an address range in order to address them. With the aid of the address, you can read from the registers or write to them. The registry access is therefore addressed.

A special feature are the registers of the input state change flags. If they are read, their data will be reset at the same time.

## **1.2. Accessing registers with 8, 16, 32 or 64 bit data width**

Accessing registers can be done in different data widths. The data can be either 8 (byte), 16 (Word), 32 (long) or 64 (eXtralong) bits wide. With the aid of the address, the desired access range is selected. An address refers to 8 bits.

If for example a 32-bit access to address 0004 hex occurs, then 4 x 8 bytes (as a data block) beginning with address 0004 hex until address 0007 hex will be read or written.



**Subdivided table in 8, 16, 32 and 64 bit register accesses**

| Access width | Address  | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
|--------------|----------|----|----|----|----|----|----|----|----|
| 8 Bit        | 0000 hex | X  |    |    |    |    |    |    |    |
| 16 Bit       | 0000 hex | X  | X  |    |    |    |    |    |    |
| 32 Bit       | 0000 hex | X  | X  | X  | X  |    |    |    |    |
| 32 Bit       | 0004 hex |    |    |    |    | X  | X  | X  | X  |
| 64 Bit       | 0000 hex | X  | X  | X  | X  | X  | X  | X  | X  |

When writing or reading, the data is written or read byte by byte. You need to pay attention to the byte order. The byte order starts with the low-byte order (byte order: Little Endian).

### Byte order of the data in the register

| Access width | Address | Value            | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
|--------------|---------|------------------|----|----|----|----|----|----|----|----|
| 8 Bit        | 04      | 1a               |    |    |    |    | 1a |    |    |    |
| 16 Bit       | 06      | 1a1b             |    |    |    |    |    |    | 1b | 1a |
| 32 Bit       | 00      | 01020304         | 04 | 03 | 02 | 01 |    |    |    |    |
| 32 Bit       | 04      | 01020304         |    |    |    |    | 04 | 03 | 02 | 01 |
| 64 Bit       | 00      | 0102030405060708 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 |

## Register assignment



## 2. Register assignment

### 2.1. General

Here you find the status register for the basic configuration. Here you can read the hardware equipment of the module.

#### 2.1.1. Reading the number of channels

| Address  | r/w  | Bit | Description   |
|----------|------|-----|---|
| FF00 hex | Read | 16  | Reading the number of digital output channels         |
| FF02 hex | Read | 16  | Reading the number of digital input channels          |
| FF04 hex | Read | 16  | Reading the number of digital in-/output channels     |
| FF06 hex | Read | 16  | Reading the number of analog output channels          |
| FF08 hex | Read | 16  | Reading the number of analog input channels           |
| FF0A hex | Read | 16  | Reading the number of stepper channels                |
| FF0E hex | Read | 16  | Reading the number of input counter channels          |
| FF10 hex | Read | 16  | Reading the number of temperature channels            |
| FF12 hex | Read | 16  | Reading the number of 48 Bit input counter channels   |
| FF14 hex | Read | 16  | Reading the number of pulse generator output channels |
| FF16 hex | Read | 16  | Reading the number of PWM output channels             |

## 2.2. Access to in-/outputs

### 2.2.1. Register overview

| Start address<br>[hex] | End address<br>[hex] | Description   |
|------------------------|----------------------|---|
| 0000                   | 000F                 | Register for digital outputs                        |
| 0020                   | 002F                 | Register for digital inputs                         |
| 0040                   | 004F                 | Register for digital input state changes            |
| 0100                   | 0101                 | Register to set direction (byte-wise)               |
| 0100                   | 011F                 | Register for digital input counter                  |
| 0120                   | 012F                 | Register to set direction (bit-wise)                |
| 01F0                   | 01F7                 | Register for digital input counter modes (RO-O8-R8) |
| 0200                   | 021F                 | Register for digital input counter (RO-O8-R8)       |
| 0800                   | 087F                 | Register for digital PWM outputs                    |
| 1000                   | 11FF                 | Register for analog inputs                          |
| 2000                   | 21FF                 | Register for analog outputs                         |

| Start address<br>[hex] | End address<br>[hex] | Description                                    |
|------------------------|----------------------|--|
| 3000                   | 31FF                 | Register for stepper                           |
| 4000                   | 40FF                 | Register for temperature inputs                |
| 5000                   | 507F                 | Register for 48 bit input counter              |
| 5800                   | 58FF                 | Register for pulse generator outputs (RO-CNT8) |

### 2.2.2. Register for digital inputs

The logical state of the voltage levels at the input modules are read out through the registers starting from address 0020 hex.

#### 2.2.2.1. Reading the inputs

| Address  | r/w  | Bit | Description                |
|----------|------|-----|----------------------------|
| 0020 hex | Read | 0-7 | Reading the inputs 1-8     |
| 0021 hex | Read | 0-7 | Reading the inputs 9-16    |
| 0022 hex | Read | 0-7 | Reading the inputs 17-24   |
| 0023 hex | Read | 0-7 | Reading the inputs 25-32   |
| 0024 hex | Read | 0-7 | Reading the inputs 33-40   |
| 0025 hex | Read | 0-7 | Reading the inputs 41-48   |
| 0026 hex | Read | 0-7 | Reading the inputs 49-56   |
| 0027 hex | Read | 0-7 | Reading the inputs 57-64   |
| 0028 hex | Read | 0-7 | Reading the inputs 65-72   |
| 0029 hex | Read | 0-7 | Reading the inputs 73-80   |
| 002A hex | Read | 0-7 | Reading the inputs 81-88   |
| 002B hex | Read | 0-7 | Reading the inputs 89-96   |
| 002C hex | Read | 0-7 | Reading the inputs 97-104  |
| 002D hex | Read | 0-7 | Reading the inputs 105-112 |
| 002E hex | Read | 0-7 | Reading the inputs 113-120 |
| 002F hex | Read | 0-7 | Reading the inputs 121-128 |

#### 2.2.2.2. Reading the input state changes

| Address  | r/w  | Bit | Description                             |
|----------|------|-----|---|
| 0040 hex | Read | 0-7 | Reading the input state changes 1-8     |
| 0041 hex | Read | 0-7 | Reading the input state changes 9-16    |
| 0042 hex | Read | 0-7 | Reading the input state changes 17-24   |
| 0043 hex | Read | 0-7 | Reading the input state changes 25-32   |
| 0044 hex | Read | 0-7 | Reading the input state changes 33-40   |
| 0045 hex | Read | 0-7 | Reading the input state changes 41-48   |
| 0046 hex | Read | 0-7 | Reading the input state changes 49-56   |
| 0047 hex | Read | 0-7 | Reading the input state changes 57-64   |
| 0048 hex | Read | 0-7 | Reading the input state changes 65-72   |
| 0049 hex | Read | 0-7 | Reading the input state changes 73-80   |
| 004A hex | Read | 0-7 | Reading the input state changes 81-88   |
| 004B hex | Read | 0-7 | Reading the input state changes 89-96   |
| 004C hex | Read | 0-7 | Reading the input state changes 97-104  |
| 004D hex | Read | 0-7 | Reading the input state changes 105-112 |
| 004E hex | Read | 0-7 | Reading the input state changes 113-120 |
| 004F hex | Read | 0-7 | Reading the input state changes 121-128 |



**WARNING:** Reading the input state changes resets them directly (to 0)

The input state change registers begins at 0040 hex.

### 2.2.2.3. Reading the input counter

| Address  | r/w  | Bit | Description                           |
|----------|------|-----|---------------------------------------|
| 0100 hex | Read | 16  | Reading the input counter of input 1  |
| 0102 hex | Read | 16  | Reading the input counter of input 2  |
| 0104 hex | Read | 16  | Reading the input counter of input 3  |
| 0106 hex | Read | 16  | Reading the input counter of input 4  |
| 0108 hex | Read | 16  | Reading the input counter of input 5  |
| 010A hex | Read | 16  | Reading the input counter of input 6  |
| 010C hex | Read | 16  | Reading the input counter of input 7  |
| 010E hex | Read | 16  | Reading the input counter of input 8  |
| 0110 hex | Read | 16  | Reading the input counter of input 9  |
| 0112 hex | Read | 16  | Reading the input counter of input 10 |
| 0114 hex | Read | 16  | Reading the input counter of input 11 |
| 0116 hex | Read | 16  | Reading the input counter of input 12 |
| 0118 hex | Read | 16  | Reading the input counter of input 13 |
| 011A hex | Read | 16  | Reading the input counter of input 14 |
| 011C hex | Read | 16  | Reading the input counter of input 15 |
| 011E hex | Read | 16  | Reading the input counter of input 16 |

#### 2.2.2.4. Reading and resetting the input counter

| Address  | r/w  | Bit | Description   |
|----------|------|-----|---|
| 0200 hex | Read | 16  | Reading and resetting the input counter of input 1  |
| 0202 hex | Read | 16  | Reading and resetting the input counter of input 2  |
| 0204 hex | Read | 16  | Reading and resetting the input counter of input 3  |
| 0206 hex | Read | 16  | Reading and resetting the input counter of input 4  |
| 0208 hex | Read | 16  | Reading and resetting the input counter of input 5  |
| 020A hex | Read | 16  | Reading and resetting the input counter of input 6  |
| 020C hex | Read | 16  | Reading and resetting the input counter of input 7  |
| 020E hex | Read | 16  | Reading and resetting the input counter of input 8  |
| 0210 hex | Read | 16  | Reading and resetting the input counter of input 9  |
| 0212 hex | Read | 16  | Reading and resetting the input counter of input 10 |

| Address  | r/w  | Bit | Description   |
|----------|------|-----|---|
| 0214 hex | Read | 16  | Reading and resetting the input counter of input 11 |
| 0216 hex | Read | 16  | Reading and resetting the input counter of input 12 |
| 0218 hex | Read | 16  | Reading and resetting the input counter of input 13 |
| 021A hex | Read | 16  | Reading and resetting the input counter of input 14 |
| 021C hex | Read | 16  | Reading and resetting the input counter of input 15 |
| 021E hex | Read | 16  | Reading and resetting the input counter of input 16 |

#### 2.2.2.5. Mode Counter-Modul (only for O8-R8)

| Address  | r/w | Bit | Description                         |
|----------|-----|-----|-------------------------------------|
| 01F0 hex | r/w | 8   | Mode of Counter-Module 0 read/write |
| 01F1 hex | r/w | 8   | Mode of Counter-Module 1 read/write |
| 01F2 hex | r/w | 8   | Mode of Counter-Module 2 read/write |
| 01F3 hex | r/w | 8   | Mode of Counter-Module 3 read/write |
| 01F4 hex | r/w | 8   | Mode of Counter-Module 4 read/write |
| 01F5 hex | r/w | 8   | Mode of Counter-Module 5 read/write |
| 01F6 hex | r/w | 8   | Mode of Counter-Module 6 read/write |
| 01F7 hex | r/w | 8   | Mode of Counter-Module 7 read/write |

#### Value for Counter-Mode

| Value | Counter   | Mode                                       |
|-------|-----------|--|
| 0     | 8x 16 Bit | 8 counter (counts only up)                 |
| 1     | 4x 16 Bit | 4 counter for Encoder (counts up and down) |

#### 2.2.2.6. Register for digital input filter

| Address  | r/w | Bit | Description                      |
|----------|-----|-----|----------------------------------|
| FD10 hex | r/w | 8   | FlipFlop input filter read/write |
| FD11 hex | r/w | 8   | Digital input filter read/write  |

The transmitted value corresponds to the time in milliseconds.

In the manual of the DELIB driver library you will find an overview table at the end of chapter 2.

This shows which module supports the above function and which values are valid.

### 2.2.3. Register for digital outputs

#### 2.2.3.1. Setting and resetting outputs

| Address  | r/w   | Bit | Description               |
|----------|-------|-----|---------------------------|
| 0000 hex | Write | 0-7 | Set/Reset outputs 1-8     |
| 0001 hex | Write | 0-7 | Set/Reset outputs 9-16    |
| 0002 hex | Write | 0-7 | Set/Reset outputs 17-24   |
| 0003 hex | Write | 0-7 | Set/Reset outputs 25-32   |
| 0004 hex | Write | 0-7 | Set/Reset outputs 33-40   |
| 0005 hex | Write | 0-7 | Set/Reset outputs 41-48   |
| 0006 hex | Write | 0-7 | Set/Reset outputs 49-56   |
| 0007 hex | Write | 0-7 | Set/Reset outputs 57-64   |
| 0008 hex | Write | 0-7 | Set/Reset outputs 65-72   |
| 0009 hex | Write | 0-7 | Set/Reset outputs 73-80   |
| 000A hex | Write | 0-7 | Set/Reset outputs 81-88   |
| 000B hex | Write | 0-7 | Set/Reset outputs 89-96   |
| 000C hex | Write | 0-7 | Set/Reset outputs 97-104  |
| 000D hex | Write | 0-7 | Set/Reset outputs 105-112 |
| 000E hex | Write | 0-7 | Set/Reset outputs 113-120 |
| 000F hex | Write | 0-7 | Set/Reset outputs 121-128 |

This register is used to set and reset digital outputs.

If the bit has a value of 0, the output is reset, if the value is 1, the output is set.

**Example:**

The value 0x02 is written to address 0x0000. Channel 2 is set, all other channels are reset.

The value 0x03 is written to address 0x0000. Channel 1 and 2 are set, all other channels are reset.

The value 0x04 is written to address 0x0000. Channel 3 is set, all other channels are reset.

The value 0x08 is written to address 0x0000. Channel 4 is set, all other channels are reset.

If you want to set or reset channels only, please use the following registers:

**Setting outputs****Resetting outputs**

The register to control the output-modules are not just writable, but also readable.

Consequently, the actual output state of one (or more modules) may be polled at a later time. Accidental termination of the software or even a crash still allows to poll the state of the output module. An ongoing process must therefore not be reset.



### 2.2.3.2. Readback outputs

| Address  | r/w  | Bit | Description              |
|----------|------|-----|--------------------------|
| 0000 hex | Read | 0-7 | Readback outputs 1-8     |
| 0001 hex | Read | 0-7 | Readback outputs 9-16    |
| 0002 hex | Read | 0-7 | Readback outputs 17-24   |
| 0003 hex | Read | 0-7 | Readback outputs 25-32   |
| 0004 hex | Read | 0-7 | Readback outputs 33-40   |
| 0005 hex | Read | 0-7 | Readback outputs 41-48   |
| 0006 hex | Read | 0-7 | Readback outputs 49-56   |
| 0007 hex | Read | 0-7 | Readback outputs 57-64   |
| 0008 hex | Read | 0-7 | Readback outputs 65-72   |
| 0009 hex | Read | 0-7 | Readback outputs 73-80   |
| 000A hex | Read | 0-7 | Readback outputs 81-88   |
| 000B hex | Read | 0-7 | Readback outputs 89-96   |
| 000C hex | Read | 0-7 | Readback outputs 97-104  |
| 000D hex | Read | 0-7 | Readback outputs 105-112 |
| 000E hex | Read | 0-7 | Readback outputs 113-120 |
| 000F hex | Read | 0-7 | Readback outputs 121-128 |

### 2.2.3.3. Setting outputs

| Address  | r/w   | Bit | Description         |
|----------|-------|-----|---------------------|
| 0080 hex | Write | 0-7 | Set outputs 1-8     |
| 0081 hex | Write | 0-7 | Set outputs 9-16    |
| 0082 hex | Write | 0-7 | Set outputs 17-24   |
| 0083 hex | Write | 0-7 | Set outputs 25-32   |
| 0084 hex | Write | 0-7 | Set outputs 33-40   |
| 0085 hex | Write | 0-7 | Set outputs 41-48   |
| 0086 hex | Write | 0-7 | Set outputs 49-56   |
| 0087 hex | Write | 0-7 | Set outputs 57-64   |
| 0088 hex | Write | 0-7 | Set outputs 65-72   |
| 0089 hex | Write | 0-7 | Set outputs 73-80   |
| 008A hex | Write | 0-7 | Set outputs 81-88   |
| 008B hex | Write | 0-7 | Set outputs 89-96   |
| 008C hex | Write | 0-7 | Set outputs 97-104  |
| 008D hex | Write | 0-7 | Set outputs 105-112 |
| 008E hex | Write | 0-7 | Set outputs 113-120 |
| 008F hex | Write | 0-7 | Set outputs 121-128 |

This register is used to set digital outputs.

If the bit has a value of 0 it is ignored, if the value is 1 the output is set.

**Example:**

The value 0x02 is written to address 0x0080. Channel 2 is set, all other channels remain unchanged.

The value 0x03 is written to address 0x0080. Channel 1 and 2 are set, all other channels remain unchanged.

The value 0x04 is written to address 0x0080. Channel 3 is set, all other channels remain unchanged.

The value 0x08 is written to address 0x0080. Channel 4 is set, all other channels remain unchanged.

If you want to set and reset channels at the same time, please use the following register:

**Setting and resetting outputs**

#### 2.2.3.4. Resetting outputs

| Address  | r/w   | Bit | Description           |
|----------|-------|-----|-----------------------|
| 00A0 hex | Write | 0-7 | Reset outputs 1-8     |
| 00A1 hex | Write | 0-7 | Reset outputs 9-16    |
| 00A2 hex | Write | 0-7 | Reset outputs 17-24   |
| 00A3 hex | Write | 0-7 | Reset outputs 25-32   |
| 00A4 hex | Write | 0-7 | Reset outputs 33-40   |
| 00A5 hex | Write | 0-7 | Reset outputs 41-48   |
| 00A6 hex | Write | 0-7 | Reset outputs 49-56   |
| 00A7 hex | Write | 0-7 | Reset outputs 57-64   |
| 00A8 hex | Write | 0-7 | Reset outputs 65-72   |
| 00A9 hex | Write | 0-7 | Reset outputs 73-80   |
| 00AA hex | Write | 0-7 | Reset outputs 81-88   |
| 00AB hex | Write | 0-7 | Reset outputs 89-96   |
| 00AC hex | Write | 0-7 | Reset outputs 97-104  |
| 00AD hex | Write | 0-7 | Reset outputs 105-112 |
| 00AE hex | Write | 0-7 | Reset outputs 113-120 |
| 00AF hex | Write | 0-7 | Reset outputs 121-128 |

This register is used to reset digital outputs.

If the bit has a value of 0 it is ignored, if the value is 1 the output is reset.

**Example:**

The value 0x02 is written to address 0x00A0. Channel 2 is reset, all other channels remain unchanged.

The value 0x03 is written to address 0x00A0. Channel 1 and 2 are reset, all other channels remain unchanged.

The value 0x04 is written to address 0x00A0. Channel 3 is reset, all other channels remain unchanged.

The value 0x08 is written to address 0x00A0. Channel 4 is reset, all other channels remain unchanged.

If you want to set and reset channels at the same time, please use the following register:

**Setting and resetting outputs**

#### 2.2.4. Register for digital PWM outputs (RO-PWM)

| Address  | r/w   | Bit | Description            |
|----------|-------|-----|------------------------|
| 0800 hex | Write | 0-7 | Setting PWM output 1   |
| 0801 hex | Write | 0-7 | Setting PWM output 2   |
| 0802 hex | Write | 0-7 | Setting PWM output 3   |
| 0803 hex | Write | 0-7 | Setting PWM output 4   |
| 0804 hex | Write | 0-7 | Setting PWM output 5   |
| 0805 hex | Write | 0-7 | Setting PWM output 6   |
| 0806 hex | Write | 0-7 | Setting PWM output 7   |
| 0807 hex | Write | 0-7 | Setting PWM output 8   |
| 0808 hex | Write | 0-7 | Setting PWM output 9   |
| 0809 hex | Write | 0-7 | Setting PWM output 10  |
| 080A hex | Write | 0-7 | Setting PWM output 11  |
| 080B hex | Write | 0-7 | Setting PWM output 12  |
| 080C hex | Write | 0-7 | Setting PWM output 13  |
| 080D hex | Write | 0-7 | Setting PWM output 14  |
| 080E hex | Write | 0-7 | Setting PWM output 15  |
| 080F hex | Write | 0-7 | Setting PWM output 16  |
| ...      | ...   | ... | ...                    |
| 087D hex | Write | 0-7 | Setting PWM output 126 |
| 087E hex | Write | 0-7 | Setting PWM output 127 |

| Address  | r/w   | Bit | Description            |
|----------|-------|-----|------------------------|
| 087F hex | Write | 0-7 | Setting PWM output 128 |

#### 2.2.4.1. Example for setting a digital PWM output

##### Example 1

Setting a PWM ratio of 50% (50% high/ 50% low) on digital PWM output 2:

##### Values of parameters

| Parameter     | Value [dec] | Value [hex] |
|---------------|-------------|-------------|
| PWM ratio [%] | 50          | 32          |
| Output 2      | 1           | 1           |

##### Content of register

| Address [hex] | Bit | Value [hex] |
|---------------|-----|-------------|
| 0801          | 0-7 | 32          |



### Example 2

Setting a PWM ratio of 20% (20% high/ 80% low) on digital PWM output 67:

#### Values of parameters

| Parameter     | Value [dec] | Value [hex] |
|---------------|-------------|-------------|
| PWM ratio [%] | 20          | 14          |
| Output 67     | 66          | 42          |

#### Content of register

| Address [hex] | Bit | Value [hex] |
|---------------|-----|-------------|
| 0842          | 0-7 | 14          |

## 2.2.5. Register for 48 bit input counter

### 2.2.5.1. 48 bit input counter address range

| Address  | Bit | Description      |
|----------|-----|------------------|
| 5000 hex | 64  | Input counter 1  |
| 5008 hex | 64  | Input counter 2  |
| 5010 hex | 64  | Input counter 3  |
| 5018 hex | 64  | Input counter 4  |
| 5020 hex | 64  | Input counter 5  |
| 5028 hex | 64  | Input counter 6  |
| 5030 hex | 64  | Input counter 7  |
| 5038 hex | 64  | Input counter 8  |
| 5040 hex | 64  | Input counter 9  |
| 5048 hex | 64  | Input counter 10 |
| 5050 hex | 64  | Input counter 11 |
| 5058 hex | 64  | Input counter 12 |
| 5060 hex | 64  | Input counter 13 |
| 5068 hex | 64  | Input counter 14 |
| 5070 hex | 64  | Input counter 15 |
| 5078 hex | 64  | Input counter 16 |

#### 2.2.5.1.1. 48 bit input counter register

| Offset<br>[hex] | r/w  | Bit   | Description          |
|-----------------|------|-------|----------------------|
| 0               | Read | 0-7   | 48 bit counter value |
| 1               | Read | 8-15  |                      |
| 2               | Read | 16-23 |                      |
| 3               | Read | 24-31 |                      |
| 4               | Read | 32-39 |                      |
| 5               | Read | 40-47 |                      |
| 6               | r/w  | 0-3   | Counter mode         |
| 6               | r/w  | 4-7   | Sub mode             |
| 7               | r/w  | 0     | Input state (0/1)    |
| 7               | -    | 1-3   | Not used             |
| 7               | r/w  | 4-7   | Input filter         |

#### 2.2.5.1.1.1. Counter mode

| Counter mode      | Value [hex] |
|-------------------|-------------|
| COUNT_RISING_EDGE | 0           |
| T                 | D           |
| FREQUENCY         | E           |
| PWM               | F           |

| Sub mode                  | Value [hex] |
|---------------------------|-------------|
| Count without reset       | 0           |
| Count with reset on read  | 1           |
| Reset, when channel 7 = 1 | 7           |
| Latch common*             | 8           |

\* The sub mode "Latch common" can be gated logically with other sub modes.

#### 2.2.5.1.1.2. Input filter

| Input filter | Value [hex] |
|--------------|-------------|
| 20 ns        | 0           |
| 100 ns       | 1           |
| 250 ns       | 2           |
| 500 ns       | 3           |
| 1 us         | 4           |
| 2,5 us       | 5           |
| 5 us         | 6           |
| 10 us        | 7           |
| 25 us        | 8           |
| 50 us        | 9           |
| 100 us       | A           |
| 250 us       | B           |
| 500 us       | C           |
| 1 ms         | D           |
| 2,5 ms       | E           |
| 5 ms         | F           |

#### 2.2.5.1.2. 48 bit input counter latch register

| Address  | Description             |
|----------|-------------------------|
| 5002 hex | Write latch common 1-8  |
| 5003 hex | Write reset common 1-8  |
| 5042 hex | Write latch common 9-16 |
| 5043 hex | Write reset common 9-16 |

## 2.2.6. Register for pulse generator outputs (RO-CNT8)

### 2.2.6.1. Pulse generator outputs address range

| Address  | Bit | Description               |
|----------|-----|---------------------------|
| 5800 hex | 128 | Pulse generator output 1  |
| 5810 hex | 128 | Pulse generator output 2  |
| 5820 hex | 128 | Pulse generator output 3  |
| 5830 hex | 128 | Pulse generator output 4  |
| 5840 hex | 128 | Pulse generator output 5  |
| 5850 hex | 128 | Pulse generator output 6  |
| 5860 hex | 128 | Pulse generator output 7  |
| 5870 hex | 128 | Pulse generator output 8  |
| 5880 hex | 128 | Pulse generator output 9  |
| 5890 hex | 128 | Pulse generator output 10 |
| 58A0 hex | 128 | Pulse generator output 11 |
| 58B0 hex | 128 | Pulse generator output 12 |
| 58C0 hex | 128 | Pulse generator output 13 |
| 58D0 hex | 128 | Pulse generator output 14 |
| 58E0 hex | 128 | Pulse generator output 15 |
| 58F0 hex | 128 | Pulse generator output 16 |

#### 2.2.6.1.1. Pulse generator output register

| Offset<br>[hex] | r/w   | Bit  | Description   |
|-----------------|-------|------|---|
| 0-3             | Write | 0-31 | Mode  |
| 4-7             | Write | 0-31 | Par0<br>(Number of pulses)<br>Par0 = 0 -> infinite pulses |
| 8-B             | Write | 0-31 | Par1<br>Low time (t [ns] / 10 - 1)                        |
| C-F             | Write | 0-31 | Par2<br>High time (t [ns] / 10 - 1)                       |

Example for setting low & high time (par1/par2)

500 ns ->  $500 / 10 - 1 = 49(\text{dec})$

7 us ->  $7000 / 10 - 1 = 699(\text{dec})$



### 2.2.6.1.2. Example for setting a pulse generator output

#### Example 1

Setting 5 pulses on pulse generator output 3 with a low time of 400ns and a high time of 600ns.

#### Values of parameters

| Parameter | Value [dec]         | Value [hex] |
|-----------|---------------------|-------------|
| Mode      | 0                   | 0           |
| Par0      | 5                   | 5           |
| Par1      | $400 / 10 - 1 = 39$ | 27          |
| Par2      | $600 / 10 - 1 = 59$ | 3B          |

### Content of register

| Address [hex] | Bit | Value [hex] |
|---------------|-----|-------------|
| 5820          | 0-7 | 00          |
| 5821          | 0-7 | 00          |
| 5822          | 0-7 | 00          |
| 5823          | 0-7 | 00          |
| 5824          | 0-7 | 05          |
| 5825          | 0-7 | 00          |
| 5826          | 0-7 | 00          |
| 5827          | 0-7 | 27          |
| 5828          | 0-7 | 00          |
| 5829          | 0-7 | 00          |
| 582A          | 0-7 | 00          |
| 582B          | 0-7 | 00          |
| 582C          | 0-7 | 3B          |
| 582D          | 0-7 | 00          |
| 582E          | 0-7 | 00          |
| 582F          | 0-7 | 00          |

### Example 2

Setting 1.000.000 pulses on pulse generator output 12 with a low time of 2,5ms and a high time of 3ms.

#### Values of parameters

| Parameter | Value [dec]                    | Value [hex] |
|-----------|--------------------------------|-------------|
| Mode      | 0                              | 0           |
| Par0      | $1.000.000 / 10 - 1 = 99.999$  | 1869F       |
| Par1      | $2.500.000 / 10 - 1 = 249.999$ | 3D08F       |
| Par2      | $3.000.000 / 10 - 1 = 299.999$ | 493DF       |

### Content of register

| Address [hex] | Bit | Value [hex] |
|---------------|-----|-------------|
| 58B0          | 0-7 | 00          |
| 58B1          | 0-7 | 00          |
| 58B2          | 0-7 | 00          |
| 58B3          | 0-7 | 00          |
| 58B4          | 0-7 | 9F          |
| 58B5          | 0-7 | 86          |
| 58B6          | 0-7 | 01          |
| 58B7          | 0-7 | 00          |
| 58B8          | 0-7 | 8F          |
| 58B9          | 0-7 | D0          |
| 58BA          | 0-7 | 03          |
| 58BB          | 0-7 | 00          |
| 58BC          | 0-7 | DF          |
| 58BD          | 0-7 | 93          |
| 58BE          | 0-7 | 04          |
| 58BF          | 0-7 | 00          |

### 2.2.7. Register to set the direction of the in-/outputs

Keep in mind, setting the direction is only usable for in-/outputs modules, whose in-/output may be changed.

The registers of the programmable in-/outputs allows to change the direction of the channels to be set as input or output. This can be done bit by bit (individual) or byte by byte (in blocks of 8 bits). By default, they are set as inputs.

#### Bit value to assign the direction

| Bit value | Description                 |
|-----------|-----------------------------|
| 0         | Set the direction as input  |
| 1         | Set the direction as output |

### 2.2.7.1. Setting byte by byte the direction of the in-/outputs

| Address  | r/w        | Bit | Description                          |
|----------|------------|-----|--------------------------------------|
| 0100 hex | Read/Write |     | Direction setting of the in-/outputs |
|          |            | 0   | In-/output 1-8                       |
|          |            | 1   | In-/output 9-16                      |
|          |            | 2   | In-/output 17-24                     |
|          |            | 3   | In-/output 25-32                     |
|          |            | 4   | In-/output 33-40                     |
|          |            | 5   | In-/output 41-48                     |
|          |            | 6   | In-/output 49-56                     |
| 0101 hex | Read/Write | 7   | In-/output 57-64                     |
|          |            |     | Direction setting of the in-/outputs |
|          |            | 0   | In-/output 65-72                     |
|          |            | 1   | In-/output 73-80                     |
|          |            | 2   | In-/output 81-88                     |
|          |            | 3   | In-/output 89-96                     |
|          |            | 4   | In-/output 97-104                    |
|          |            | 5   | In-/output 105-112                   |
|          |            | 6   | In-/output 113-120                   |
|          |            | 7   | In-/output 121-128                   |

Example to set in-/outputs to the following direction:

| Direction | r/w   | In-/output number | Bit value |
|-----------|-------|-------------------|-----------|
| Input     | Read  | 9-16              | 0         |
| Input     | Read  | 41-48             | 0         |
| Output    | Write | 49-56             | 1         |

In Register 0100 hex, the 2nd bit is to be set to 0, the 5th bit is to be set to 1 and the 6th bit is to be set to 1. The resulting bit pattern is: x10x xx0x, while "x" stays for the value, that is already in the register.

### 2.2.7.2. Setting bit by bit the direction of the in-/outputs

| Address  | r/w   | Bit | Description                              |
|----------|-------|-----|--|
| 0120 hex | Write | 0-7 | Setting direction of in-/outputs 1-8     |
| 0121 hex | Write | 0-7 | Setting direction of in-/outputs 9-16    |
| 0122 hex | Write | 0-7 | Setting direction of in-/outputs 17-24   |
| 0123 hex | Write | 0-7 | Setting direction of in-/outputs 25-32   |
| 0124 hex | Write | 0-7 | Setting direction of in-/outputs 33-40   |
| 0125 hex | Write | 0-7 | Setting direction of in-/outputs 41-48   |
| 0126 hex | Write | 0-7 | Setting direction of in-/outputs 49-56   |
| 0127 hex | Write | 0-7 | Setting direction of in-/outputs 57-64   |
| 0128 hex | Write | 0-7 | Setting direction of in-/outputs 65-72   |
| 0129 hex | Write | 0-7 | Setting direction of in-/outputs 73-80   |
| 012A hex | Write | 0-7 | Setting direction of in-/outputs 81-88   |
| 012B hex | Write | 0-7 | Setting direction of in-/outputs 89-96   |
| 012C hex | Write | 0-7 | Setting direction of in-/outputs 97-104  |
| 012D hex | Write | 0-7 | Setting direction of in-/outputs 105-112 |
| 012E hex | Write | 0-7 | Setting direction of in-/outputs 113-120 |
| 012F hex | Write | 0-7 | Setting direction of in-/outputs 121-128 |



**Example to set the direction of the first 8 in-/outputs as follows:**

Channel 1: input

Channel 2: input

Channel 3: output

Channel 4: output

Channel 5: input

Channel 6: input

Channel 7: output

Channel 8: output

The address 0120 hex is to be set to 11001100.

## 2.2.8. Register for analog inputs (A/D)

### 2.2.8.1. A/D converter address range

| Address [hex] | Description              |
|---------------|--------------------------|
| 1000-1003     | Accessing A/D channel 1  |
| 1004-1007     | Accessing A/D channel 2  |
| 1008-100B     | Accessing A/D channel 3  |
| 100C-100F     | Accessing A/D channel 4  |
| 1010-1013     | Accessing A/D channel 5  |
| 1014-1017     | Accessing A/D channel 6  |
| 1018-101B     | Accessing A/D channel 7  |
| 101C-101F     | Accessing A/D channel 8  |
| 1020-1023     | Accessing A/D channel 9  |
| 1024-1027     | Accessing A/D channel 10 |
| 1028-102B     | Accessing A/D channel 11 |
| 102C-102F     | Accessing A/D channel 12 |
| 1030-1033     | Accessing A/D channel 13 |
| 1034-1037     | Accessing A/D channel 14 |
| 1038-103B     | Accessing A/D channel 15 |
| 103C-103F     | Accessing A/D channel 16 |
| ...           | ...                      |

| Address [hex] | Description               |
|---------------|---------------------------|
| 10F8-10FB     | Accessing A/D channel 63  |
| 10FC-10FF     | Accessing A/D channel 64  |
| 1100-1103     | Accessing A/D channel 65  |
| ...           | ...                       |
| 1178-117B     | Accessing A/D channel 95  |
| 117C-117F     | Accessing A/D channel 96  |
| 1180-1183     | Accessing A/D channel 97  |
| ...           | ...                       |
| 11F4-11F7     | Accessing A/D channel 126 |
| 11F8-11FB     | Accessing A/D channel 127 |
| 11FC-11FF     | Accessing A/D channel 128 |

### 2.2.8.2. A/D converter register

| Offset [hex] | r/w          | Bit  | Description                               |
|--------------|--------------|------|---|
| 0, 1         | Read         | 0-15 | Reading A/D value                         |
| 2            | Write        | 0-7  | Not used                                  |
| 3            | Read / Write | 0-7  | "A/D mode" writing/polling (reading back) |

#### 2.2.8.2.1. A/D - mode

| Mode [hex] | Description                                       |
|------------|---|
| 00         | 0-10V   |
| 01         | 0-5V  |
|            |   |
| 40         | $\pm 10V$   |
| 41         | $\pm 5V$  |
|            |   |
| 80         | 0..20 mA (only available for RO-AD16 with I-mode) |
| 81         | 4..20 mA (only available for RO-AD16 with I-mode) |
| 82         | 0..24 mA (only available for RO-AD16 with I-mode) |

## 2.2.9. Register for temperature inputs

### 2.2.9.1. Temperature input address range

| Address [hex] | Description                      |
|---------------|----------------------------------|
| 3000-3007     | Accessing temperature channel 1  |
| 3008-300F     | Accessing temperature channel 2  |
| 3010-3017     | Accessing temperature channel 3  |
| 3018-301F     | Accessing temperature channel 4  |
| 3020-3027     | Accessing temperature channel 5  |
| 3028-302F     | Accessing temperature channel 6  |
| 3030-3037     | Accessing temperature channel 7  |
| 3038-303F     | Accessing temperature channel 8  |
| 3040-3047     | Accessing temperature channel 9  |
| 3048-304F     | Accessing temperature channel 10 |
| 3050-3057     | Accessing temperature channel 11 |
| 3058-305F     | Accessing temperature channel 12 |
| 3060-3067     | Accessing temperature channel 13 |
| 3068-306F     | Accessing temperature channel 14 |
| 3070-3077     | Accessing temperature channel 15 |
| 3078-307F     | Accessing temperature channel 16 |

| Address [hex] | Description                      |
|---------------|----------------------------------|
| 3080-3087     | Accessing temperature channel 17 |
| 3088-308F     | Accessing temperature channel 18 |
| 3090-3097     | Accessing temperature channel 19 |
| ...           | ...                              |
| 30C8-30CF     | Accessing temperature channel 26 |
| 30D0-30D7     | Accessing temperature channel 27 |
| 30D8-30DF     | Accessing temperature channel 28 |
| 30E0-30E7     | Accessing temperature channel 29 |
| 30E8-30EF     | Accessing temperature channel 30 |
| 30F0-30F7     | Accessing temperature channel 31 |
| 30F8-30FF     | Accessing temperature channel 32 |

#### 2.2.9.1.1. Temperature input register

| Address [hex] | r/w  | Bit  | Description  |
|---------------|------|------|--|
| base + 0, 1   | Read | 0-14 | Read bit 0-14 of the current temperature (value = temp*factor)   |
|               | Read | 15   | Algebraic sign (0 = positive, 1 = negative)  |
| base + 2      | Read | 0-7  | Factor (<br>0 [dec] = illegal value / sensor not connected,<br>1 [dec] = factor 10,<br>2 [dec] = factor 100) |
| base + 3      | Read | 0    | State PT100 sensor<br>(1 = connected, 0 = not connected)   |

Calculation of temperature

Temperature = (algebraic sign) value [dec] / factor

### 2.2.9.1.2. Example for getting temperatures

#### Example 1

Getting the temperature of temperature input 3:

| Address [hex] | Bit  | Value [dec] | Description               |
|---------------|------|-------------|---------------------------|
| 4010, 4011    | 0-14 | 123         | Decimal value 123         |
|               | 15   | 0           | Algebraic sign (positive) |
| 4012          | 0-7  | 1           | Factor 10                 |
| 4013          | 0    | 1           | PT100 sensor connected    |

Temperature =  $+123 / 10 = +12,3\text{ °C}$



## Example 2

Getting the temperature of temperature input 1:

| Address [hex] | Bit  | Value [dec] | Description               |
|---------------|------|-------------|---------------------------|
| 4010, 4011    | 0-14 | 2263        | Decimal value 2263        |
|               | 15   | 1           | Algebraic sign (negative) |
| 4012          | 0-7  | 2           | Factor 100                |
| 4013          | 0    | 1           | PT100 sensor connected    |

Temperature =  $-2263 / 100 = -22,63\text{ °C}$

## 2.2.10. Register for analog outputs (D/A)

### 2.2.10.1. D/A converter address range

| Address [hex] | Description              |
|---------------|--------------------------|
| 2000-2007     | Accessing D/A channel 1  |
| 2008-200F     | Accessing D/A channel 2  |
| 2010-2017     | Accessing D/A channel 3  |
| 2018-201F     | Accessing D/A channel 4  |
| 2020-2027     | Accessing D/A channel 5  |
| 2028-202F     | Accessing D/A channel 6  |
| 2030-2037     | Accessing D/A channel 7  |
| 2038-203F     | Accessing D/A channel 8  |
| 2040-2047     | Accessing D/A channel 9  |
| 2048-204F     | Accessing D/A channel 10 |
| 2050-2057     | Accessing D/A channel 11 |
| 2058-205F     | Accessing D/A channel 12 |
| 2060-2067     | Accessing D/A channel 13 |
| 2068-206F     | Accessing D/A channel 14 |
| 2070-2077     | Accessing D/A channel 15 |
| 2078-207F     | Accessing D/A channel 16 |
| ...           | ...                      |

| Address [hex] | Description              |
|---------------|--------------------------|
| 20F0-20F7     | Accessing D/A channel 31 |
| 20F8-20FF     | Accessing D/A channel 32 |
| 2100-2107     | Accessing D/A channel 33 |
| ...           | ...                      |
| 2170-2177     | Accessing D/A channel 47 |
| 2178-217F     | Accessing D/A channel 48 |
| 2180-2187     | Accessing D/A channel 49 |
| ...           | ...                      |
| 21E8-21EF     | Accessing D/A channel 62 |
| 21F0-21F7     | Accessing D/A channel 63 |
| 21F8-21FF     | Accessing D/A channel 64 |

### 2.2.10.2. D/A converter register

| Offset<br>[hex] | r/w        | Bit  | Description  |
|-----------------|------------|------|--|
| 0, 1            | Write/Read | 0-15 | Setting D/A value (register pollable)  |
| 2               | Write/Read | 0-7  | "D/A – mode" writing/polling (reading back)  |
| 3               | Write/Read | 0-7  | Output enable (0=enable / 1=disable)   |
| 4               | Write/Read | 0-7  | Slew rate step (RO-DA2-ISO only)   |
| 5               | Write/Read | 0-7  | Slew rate Update Clock (RO-DA2-ISO only)   |
| 6               | Write/Read | 0-7  | Slew rate enable (1=enable / 0=disable)<br>(RO-DA2-ISO only)   |
| 7               | Write      | 0-7  | Save configuration as default into EEPROM<br>(with data = 10 [hex])  |
| 7               | Write      | 0-7  | Load configuration from EEPROM<br>(with data = 11 [hex])   |
| 7               | Write      | 0-7  | Load factory-default configuration<br>(with data = 12 [hex])   |
| 7               | Read       | 0-7  | Last D/A activity can be read back.<br>01 = D/A data has been written<br>10 = Default-configuration has been saved into the EEPROM<br>11 = Configuration from EEPROM has been loaded<br>12 = Factory-default configuration has been loaded |

#### 2.2.10.2.1. D/A - mode

| Mode [hex] | Description                |
|------------|----------------------------|
| 00         | 0-10V                      |
| 01         | 0-5V                       |
|            |                            |
| 40         | +10V                       |
| 41         | +5V                        |
|            |                            |
| 80         | 0..20 mA (RO-DA2-ISO only) |
| 81         | 4..20 mA (RO-DA2-ISO only) |
| 82         | 0..24 mA (RO-DA2-ISO only) |

## 2.2.11. Register for stepper

### 2.2.11.1. Stepper address range

| Address [hex] | Description         |
|---------------|---------------------|
| 3000-303F     | Accessing Stepper 1 |
| 3040-307F     | Accessing Stepper 2 |
| 3080-30BF     | Accessing Stepper 3 |
| 30C0-30FF     | Accessing Stepper 4 |
| 3100-313F     | Accessing Stepper 5 |
| 3140-317F     | Accessing Stepper 6 |
| 3180-31BF     | Accessing Stepper 7 |
| 31C0-31FF     | Accessing Stepper 8 |

### 2.2.11.2. Stepper command register

| Address [hex] | Parameter | Bit   | Description                    |
|---------------|-----------|-------|--------------------------------|
| base + 0010   | par 3     | 0-7   | Read/Write Bit 0-7 of par 3    |
| base + 0011   | par 3     | 8-15  | Read/Write Bit 8-15 of par 3   |
| base + 0012   | par 3     | 16-23 | Read/Write Bit 16-23 of par 3  |
| base + 0013   | par 3     | 24-31 | Read/Write Bit 24-31 of par 3  |
|               |           |       |                                |
| base + 0014   | par 2     | 0-7   | Read/Write Bit 0-7 of par 2    |
| base + 0015   | par 2     | 8-15  | Read/Write Bit 8-15 of par 2   |
| base + 0016   | par 2     | 16-23 | Read/Write Bit 16-23 of par 2  |
| base + 0017   | par 2     | 24-31 | Read/Write Bit 24-31 of par 2  |
|               |           |       |                                |
| base + 0018   | par 1     | 0-7   | Read/Write Bit 0-7 of par 1    |
| base + 0019   | par 1     | 8-15  | Read/Write Bit 8-15 of par 1   |
| base + 001A   | par 1     | 16-23 | Read/Write Bit 16-23 of par 1  |
| base + 001B   | par 1     | 24-31 | Read/Write Bit 24-31 of par 1  |
|               |           |       |                                |
| base + 001C   | command   | 0-7   | Read/Write Bit 0-7 of command  |
| base + 001D   | command   | 8-15  | Read/Write Bit 8-15 of command |

| Address [hex] | Parameter | Bit   | Description                     |
|---------------|-----------|-------|---------------------------------|
| base + 001E   | command   | 16-23 | Read/Write Bit 16-23 of command |
| base + 001F   | command   | 24-31 | Read/Write Bit 24-31 of command |

#### 2.2.11.2.1. Command list

| Command with DAPI_STEPPER_CMD_ | Value (hex) | Description  |
|--------------------------------|-------------|--|
| SET_MOTORCHARACTERISTIC        | 1 (hex)     | Setting the motor configuration                      |
| GET_MOTORCHARACTERISTIC        | 2 (hex)     | Getting the motor configuration                      |
| SET_POSITION                   | 3 (hex)     | Setting the motor position                           |
| GO_POSITION                    | 4 (hex)     | Going to a position                                  |
| GET_POSITION                   | 5 (hex)     | Getting the position                                 |
| SET_FREQUENCY                  | 6 (hex)     | Setting the motor reference frequency                |
| SET_FREQUENCY_DIRECTLY         | 7 (hex)     | Setting the motor frequency                          |
| GET_FREQUENCY                  | 8 (hex)     | Getting the motor frequency                          |
| FULLSTOP                       | 9 (hex)     | Stopping the motor immediately                       |
| STOP                           | 10 (hex)    | Stopping the motor (deceleration slope will be used) |
| GO_REFSWITCH                   | 11 (hex)    | Going to a reference position                        |



| Command<br>DAPI_STEPPER_CMD_     | with<br>Value<br>(hex) | Description  |
|----------------------------------|------------------------|--|
| DISABLE                          | 14<br>(hex)            | En-/disabling the motor                            |
| MOTORCHARACTERISTIC_LOAD_DEFAULT | 15<br>(hex)            | Setting the motor characteristic to default        |
| MOTORCHARACTERISTIC_EEPROM_SAVE  | 16<br>(hex)            | Saving the motor characteristic to EEPROM          |
| MOTORCHARACTERISTIC_EEPROM_LOAD  | 17<br>(hex)            | Loading the motor characteristic out of the EEPROM |
| GET_CPU_TEMP                     | 18<br>(hex)            | Getting the temperature of the CPU                 |
| GET_MOTOR_SUPPLY_VOLTAGE         | 19<br>(hex)            | Getting the supply voltage of the motor            |
| GO_POSITION_RELATIVE             | 20<br>(hex)            | Going to a relative position                       |

#### 2.2.11.2.2. Explanation of par 1, 2 and 3

You can take the values for par1, par2, par3 out of the DELIB manual(go to DELIB manual).

#### 2.2.11.3. Reading stepper status

| Address<br>[hex] | Parameter<br>DAPI_STEPPER_<br>STATUS_ | Bit   | Description                           |
|------------------|---------------------------------------|-------|---------------------------------------|
| base + 0028      | GET_POSITION                          | 0-7   | Reading bit 0-7 of a position         |
| base + 0029      | GET_POSITION                          | 8-15  | Reading bit 8-15 of a position        |
| base + 002A      | GET_POSITION                          | 16-23 | Reading bit 16-23 of a position       |
| base + 002B      | GET_POSITION                          | 24-31 | Reading bit 24-31 of a position       |
|                  |                                       |       |                                       |
| base + 002E      | GET_ACTIVITY                          | 0-7   | Reading bit 0-7 of status information |
| base + 002F      | GET_SWITCH                            | 0-7   | Reading bit 0-7 of switch status      |

##### 2.2.11.3.1. Explanation of command GET\_POSITION

With this command, the motor position can be read in 1/16 step units.

#### 2.2.11.3.2. Explanation of the bits for command GET\_ACTIVITY

| Bit | Command            | Description                       |
|-----|--------------------|-----------------------------------|
| 0   | DISABLE            | Motor is disabled                 |
| 1   | MOTORSTROMACTIV    | Motor phase current is active     |
| 2   | HALTESTROMACTIV    | Hold phase current is active      |
| 3   | GOPOSITIONACTIV    | GoPosition is active              |
| 4   | GOPOSITIONBREMSSEN | GoPosition deceleration is active |
| 5   | GOREFERENZACTIV    | GoPosition deceleration is active |
| 6   | not used           | not used                          |
| 7   | not used           | not used                          |

#### 2.2.11.3.3. Explanation of the bits for command GET\_SWITCH

| Bit | Switch            | Description                 |
|-----|-------------------|-----------------------------|
| 0   | Endswitch 1       | Endswitch 1 is closed       |
| 1   | Endswitch 2       | Endswitch 2 is closed       |
| 2   | Referenceswitch 1 | Referenceswitch 1 is closed |
| 3   | Referenceswitch 2 | Referenceswitch 2 is closed |
| 4   | not used          | not used                    |
| 5   | not used          | not used                    |
| 6   | not used          | not used                    |
| 7   | not used          | not used                    |

## 2.2.12. Timeout-register for digital and analog outputs (D/A)

With an active timeout, the outputs are switched off, if the data transfer was interrupted for at least a certain time. Further operation of the connected equipment and hence potential damage will be prevented. This could e.g. occur during unintended software termination, software crash or releasing of the data cable.

### 2.2.12.1. Timeout-register

A timeout-event occurs, if the timeout is activated and no data is transmitted for at least the duration of the timeout-value.

| Address<br>[hex] | r/w   | Bit | Description  |
|------------------|-------|-----|--|
| FD00             | Write | 0-7 | Timeout enable (1 = enable / 0 = disable)            |
| FD01             | Read  | 0-7 | Read, if a timeout-value occurred (0 = no / 1 = yes) |

#### 2.2.12.1.1. Timeout address range for timeout-value

| Address<br>[hex] | r/w   | Bit  | Description                             |
|------------------|-------|------|---|
| FD02-FD03        | Write | 0-15 | Sets the timeout-value (value * 100 ms) |

The timeout-value is the product of 16-bit-value \* 100 ms (the unit is therefore 100 ms).

# CAN protocol



## 3. CAN protocol

### 3.1. Speed and interface parameters

With the CAN Bus, a data transfer rate of 10Kbit/sec up to 1Mbit/sec may be reached. Other transfer rates may be set up using a DIP switch.

### 3.2. Structure of the "sending packet"

The first byte of a packet is the "COMMAND".

| Byte no. | Description | Explanation                                       |
|----------|-------------|---|
| 1. byte  | COMMAND     | COMMAND indicates the command and the bit width.  |
| 2. byte  | ADDRESS     | bit 0-7   |
| 3. byte  | ADDRESS     | bit 8-15  |
| 4. byte  | JOB-ID      | The JOB-ID labels the packet with a stamp number. |
| 5. byte  | DATA        | bit 0-7   |
| 6. byte  | DATA        | bit 8-15  |
| 7. byte  | DATA        | bit 16-23   |
| 8. byte  | DATA        | bit 24-31   |

### 3.2.1. COMMAND Write

The COMMAND write indicates that the tokens "0x20" are used to write to the addresses. The data width can be determined.

| COMMAND | Data width (bits)                  |
|---------|------------------------------------|
|         | data is transmitted to the device. |
| 0x21    | bit (1 bit)                        |
| 0x22    | byte (8 bit)                       |
| 0x23    | word (16 bit)                      |
| 0x24    | long (32 bit)                      |



### 3.2.2. COMMAND Read

The COMMAND read indicates that the tokens "0x10" are used to read from the addresses. The data width can be determined.

| COMMAND | Data width (bits)   |
|---------|---|
|         | No data is transmitted to the device. The device sends back data as response. |
| 0x11    | bit (1 bit)   |
| 0x12    | byte (8 bit)  |
| 0x13    | word (16 bit)   |
| 0x14    | long (32 bit)   |

### 3.2.3. ADDRESS

An address is 16 bit wide and is transferred as 2 hexadecimal values.

### 3.2.4. JOB-ID

The JOB-ID indicates the actual sending packet. Two consecutive sending packets may not have the same JOB-ID.

#### HINT:

After a successful data transfer, the sending routine should increase the JOB-ID by "1". That way, the successive numbers "0", then "1" .. "255" and again "0" are sent.

### 3.2.5. DATA

When a write command is used, the data field is present. A read command does not have any data field.

## 3.3. Example for a write command

In the following example, the data word 0F hex is written to the address 0012 hex. Here, the JOB-ID is 0x34.

| Byte no. | Description | Bits | Data [hex] |
|----------|-------------|------|------------|
| 1. byte  | COMMAND     | 0-7  | 22         |
| 2. byte  | ADDRESS     | 0-7  | 12         |
| 3. byte  | ADDRESS     | 8-15 | 00         |
| 4. byte  | JOB-ID      | 0-7  | 34         |

| Byte no. | Description | Bits  | Data [hex] |
|----------|-------------|-------|------------|
| 5. byte  | DATA        | 0-7   | 0F         |
| 6. byte  | DATA        | 8-15  | 00         |
| 7. byte  | DATA        | 16-23 | 00         |
| 8. byte  | DATA        | 24-31 | 00         |

### 3.4. Example for writing the first 16 relays

Relays 1-13 are to be switched off. Relay 14 is to be switched on and relay 15 and 16 are to be switched off.

| Byte no. | Description | Bits  | Data [hex] |
|----------|-------------|-------|------------|
| 1. byte  | COMMAND     | 0-7   | 23         |
| 2. byte  | ADDRESS     | 0-7   | 00         |
| 3. byte  | ADDRESS     | 8-15  | 00         |
| 4. byte  | JOB-ID      | 0-7   | 35         |
| 5. byte  | DATA        | 0-7   | 00         |
| 6. byte  | DATA        | 8-15  | 20         |
| 7. byte  | DATA        | 16-23 | 00         |
| 8. byte  | DATA        | 24-31 | 00         |

### 3.5. Structure of the "response packet"

The response packet consists of a 7-byte CAN-packet. Such a packet is build up of a response, a validation checksum of the received data, a JOB-ID and 4 DATA bytes.

#### 3.5.1. "OK" - response packet

This means the response is "ok" and informs the sender that the write command is successfully executed.

OK-response of the sending packet example from chapter 3.2

| Byte no. | Description   | Signal (Bits)         | Modus  |
|----------|---|-----------------------|--|
| 1. byte  | response.   | 1                     | OK   |
| 2. byte  | Checksum to check the validity of the received data | 0<br>1<br>2<br>3<br>4 | No data<br>1 bit<br>1 byte<br>1 word<br>1 long |
| 3. byte  | JOB-ID  |                       |  |
| 4. byte  | DATA  | 0-7                   |  |
| 5. byte  | DATA  | 8-15                  |  |
| 6. byte  | DATA  | 16-23                 |  |
| 7. byte  | DATA  | 24-31                 |  |

### 3.5.2. "Error" - response packet

#### 3.5.2.1. Error code "0xfd" – JOB-ID double

A response packet beginning with a "0xfd" indicates that the JOB-ID already exists.

| Byte no. | Description                                 |
|----------|---|
| 1        | "0xfd" indicates the JOB-ID already exists. |
| 2        | JOB-ID                                      |

Command of the sending packet and the "0xfd" - response

| Read request | Response data     |                 |
|--------------|-------------------|-----------------|
| COMMAND      | Data width (bits) | Amount of bytes |
| 0x11         | 1                 | 1               |
| 0x12         | 8                 | 1               |
| 0x13         | 16                | 2               |
| 0x14         | 32                | 4               |

### 3.5.2.2. Error code "0xfe" – invalid command

The last sent "sending packet" has an invalid command.

#### "0xfe" - format

| Byte no. | Description                 |
|----------|-----------------------------|
| 1        | "0xfe" for invalid command. |

## 3.6. Structure of the CAN TX/RX packages

The following pages shows the structure of the TX/RX CAN packages of the I/O modules. The pages also show how to calculate the A/D and D/A values.

### 3.6.1. Digital inputs

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                    |
|---------------|----------------------------|
| 1             | DI channel 1-8 (Bit 0-7)   |
| 2             | DI channel 9-16 (Bit 0-7)  |
| 3             | DI channel 17-24 (Bit 0-7) |
| 4             | DI channel 25-32 (Bit 0-7) |
| 5             | DI channel 33-40 (Bit 0-7) |
| 6             | DI channel 41-48 (Bit 0-7) |
| 7             | DI channel 49-56 (Bit 0-7) |
| 8             | DI channel 57-64 (Bit 0-7) |



### 3.6.2. Digital outputs

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                    |
|---------------|----------------------------|
| 1             | DO channel 1-8 (Bit 0-7)   |
| 2             | DO channel 9-16 (Bit 0-7)  |
| 3             | DO channel 17-24 (Bit 0-7) |
| 4             | DO channel 25-32 (Bit 0-7) |
| 5             | DO channel 33-40 (Bit 0-7) |
| 6             | DO channel 41-48 (Bit 0-7) |
| 7             | DO channel 49-56 (Bit 0-7) |
| 8             | DO channel 57-64 (Bit 0-7) |

### 3.6.3. Digital input counter (16-bit)

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                 |
|---------------|-------------------------|
| 1             | DI counter 1 (Bit 0-7)  |
| 2             | DI counter 1 (Bit 8-15) |
| 3             | DI counter 2 (Bit 0-7)  |
| 4             | DI counter 2 (Bit 8-15) |
| 5             | DI counter 3 (Bit 0-7)  |
| 6             | DI counter 3 (Bit 8-15) |
| 7             | DI counter 4 (Bit 0-7)  |
| 8             | DI counter 4 (Bit 8-15) |

### 3.6.4. Digital input counter (48-Bit) - 32-Bit package

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                    |
|---------------|----------------------------|
| 1             | CNT8 counter 1 (Bit 0-7)   |
| 2             | CNT8 counter 1 (Bit 8-15)  |
| 3             | CNT8 counter 1 (Bit 16-23) |
| 4             | CNT8 counter 1 (Bit 24-31) |
| 5             | CNT8 counter 2 (Bit 0-7)   |
| 6             | CNT8 counter 2 (Bit 8-15)  |
| 7             | CNT8 counter 2 (Bit 16-23) |
| 8             | CNT8 counter 2 (Bit 24-31) |

Note: Only the first 32-Bit of each 48-Bit input counter can be send automatically.

### 3.6.5. Digital input counter (48-Bit) - 64-Bit package

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Bit | Content                           |
|---------------|-----|-----------------------------------|
| 1             | 0-7 | CNT8 counter 1 (Bit 0-7)          |
| 2             | 0-7 | CNT8 counter 1 (Bit 8-15)         |
| 3             | 0-7 | CNT8 counter 1 (Bit 16-23)        |
| 4             | 0-7 | CNT8 counter 1 (Bit 24-31)        |
| 5             | 0-7 | CNT8 counter 1 (Bit 31-39)        |
| 6             | 0-7 | CNT8 counter 1 (Bit 40-47)        |
| 7             | 0-3 | CNT8 counter 1 Counter mode       |
| 7             | 4-7 | CNT8 counter 1 Submode            |
| 8             | 0   | CNT8 counter 1 Input status (0/1) |
| 8             | 1-3 | Not used                          |
| 8             | 4-7 | CNT8 counter 1 Input filter       |

### 3.6.6. Analog in-/outputs

#### 3.6.6.1. Analog inputs

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                  |
|---------------|--------------------------|
| 1             | A/D channel 5 (Bit 0-7)  |
| 2             | A/D channel 5 (Bit 8-15) |
| 3             | A/D channel 6 (Bit 0-7)  |
| 4             | A/D channel 6 (Bit 8-15) |
| 5             | A/D channel 7 (Bit 0-7)  |
| 6             | A/D channel 7 (Bit 8-15) |
| 7             | A/D channel 8 (Bit 0-7)  |
| 8             | A/D channel 8 (Bit 8-15) |

The range of A/D values e.g. 0-5V sets the digital values. The range can be configured in the CAN-Configuration Utility.

**Note:**

The hex value FFFF stands for the highest value from a range and the hex value 0000 stands for the lowest value from a range.

Formula (A/D-Mode +/-10V, +/-5V or +/-2,5V)

Voltage =  $(\text{Value} * (\text{max. Voltage} * 2) / 0xFFFF) - \text{max. Voltage}$

Formula (A/D-Modus 0..10V, 0..5V or 0..2,5V)

Voltage =  $\text{Value} * \text{max. Voltage} / 0xFFFF$

Formula (A/D-Modus 0..20mA, 4..20mA or 0..24mA)

Current =  $\text{Value} * 25 / 0xFFFF$

### 3.6.6.2. Analog outputs

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                  |
|---------------|--------------------------|
| 1             | D/A channel 1 (Bit 0-7)  |
| 2             | D/A channel 1 (Bit 8-15) |
| 3             | D/A channel 2 (Bit 0-7)  |
| 4             | D/A channel 2 (Bit 8-15) |
| 5             | D/A channel 3 (Bit 0-7)  |
| 6             | D/A channel 3 (Bit 8-15) |
| 7             | D/A channel 4 (Bit 0-7)  |
| 8             | D/A channel 4 (Bit 8-15) |

The range of A/D values e.g. 0-5V sets the digital values. The range can be configured in the CAN-Configuration Utility.

**Note:**

The hex value FFFF stands for the highest value from a range and the hex value 0000 stands for the lowest value from a range.

### 3.6.6.3. Examples

#### Example Voltage range $\pm 10\text{V}$

| Value (hex) | Voltage |
|-------------|---------|
| FFFF        | +10 V   |
| 8000        | 0 V     |
| 0000        | -10V    |

#### Example:

CAN-Data-Byte0, 1 = 0x4711[hex]

Voltage range = +/-10 V

#### Calculation:

Voltage =  $(0x4711 * (10 * 2) / 0xFFFF) - 10 = -4,45 \text{ V}$



#### Example Voltage range 0-5V

| Value (hex) | Voltage |
|-------------|---------|
| FFFF        | +5 V    |
| 8000        | +2,5 V  |
| 0000        | 0 V     |

#### Example:

CAN-Data-Byte0, 1 = 0x4711[hex]

Voltage range = 0-5 V

#### Calculation:

Voltage =  $0x4711 * 5 / 0xFFFF = 1,38 \text{ V}$

### Example current range $\pm 10V$

| Value (hex) | Current |
|-------------|---------|
| FFFF        | 25 mA   |
| 8000        | 12,5 mA |
| 0000        | 0 mA    |

### Example:

CAN-Data-Byte0, 1 = 0x4711[hex]

Current range = 0..20 mA, 4..20 mA oder 0..24 mA

### Calculation:

Current =  $0x4711 * 25 / 0xFFFF = 6,94 \text{ mA}$

### Note:

Auto-TX packages with a current calculates the value always with 0-25mA range.

### 3.6.7. Temperature inputs

#### Package structure for 2 PT100 channels

| CAN-Data-Byte | Bit | Content  |
|---------------|-----|--|
| 1             | 0-7 | Value channel 1 (Bit 0-7)  |
| 2             | 0-6 | Value channel 1 (Bit 8-14)   |
|               | 7   | Sign channel 1 (0 = positive, 1 = negative)  |
| 3             | 0-1 | Factor channel 1 (<br>0 [dec] = illegal value / sensor not connected,<br>1 [dec] = Factor 10,<br>2 [dec] = Factor 100) |
|               | 2-7 | not used   |
| 4             | 0   | Status PT100 sensor channel 1<br>(0 = not connected, 1 = connected)  |
|               | 1-7 | not used   |
| 5             | 0-7 | Value channel 2 (Bit 0-7)  |
| 6             | 0-6 | Value channel 2 (Bit 8-14)   |
|               | 7   | Sign channel 2 (0 = positive, 1 = negative)  |
| 7             | 0-1 | Factor channel 2 (<br>0 [dec] = illegal value / sensor not connected,<br>1 [dec] = Factor 10,<br>2 [dec] = Factor 100) |
|               | 2-7 | not used   |
| 8             | 0   | Status PT100 sensor channel 2<br>(0 = not connected, 1 = connected)  |

| CAN-Data-Byte | Bit | Content  |
|---------------|-----|----------|
|               | 0-7 | not used |

Calculation of the temperature

Temperature = (Sign) Value [dez] / Factor

### 3.6.8. Stepper

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content          |
|---------------|------------------|
| 1             | COMMAND          |
| 2             | PAR1 (Bit 0-7)   |
| 3             | PAR1 (Bit 8-15)  |
| 4             | PAR1 (Bit 16-23) |
| 5             | PAR1 (Bit 24-31) |
| 6             | PAR2 (Bit 0-7)   |
| 7             | PAR2 (Bit 8-15)  |
| 8             | PAR3 (Bit 0-7)   |

### 3.6.8.1. Command-List

| Kommando<br>DAPI_STEPPER_CMD_    | mit<br>Wert<br>(hex) | Bedeutung                          |
|----------------------------------|----------------------|------------------------------------|
| SET_MOTORCHARACTERISTIC          | 1 (hex)              | Write motor configuration          |
| GET_MOTORCHARACTERISTIC          | 2 (hex)              | Read motor configuration           |
| SET_POSITION                     | 3 (hex)              | Set motor position                 |
| GO_POSITION                      | 4 (hex)              | Drive to specific position         |
| GET_POSITION                     | 5 (hex)              | Read motor position                |
| SET_FREQUENCY                    | 6 (hex)              | Set motor frequency                |
| SET_FREQUENCY_DIRECTLY           | 7 (hex)              | Set motor frequency directly       |
| GET_FREQUENCY                    | 8 (hex)              | Read motor frequency               |
| FULLSTOP                         | 9 (hex)              | Full stop                          |
| STOP                             | 10<br>(hex)          | Stop with braking ramp             |
| GO_REFSWITCH                     | 11<br>(hex)          | Drive to reference position        |
| DISABLE                          | 14<br>(hex)          | turn motor on/off                  |
| MOTORCHARACTERISTIC_LOAD_DEFAULT | 15<br>(hex)          | Set motor configuration to default |
| MOTORCHARACTERISTIC_EEPROM_SAVE  | 16<br>(hex)          | Save motor configuration to EEPROM |

| Kommando<br>DAPI stepper_CMD_   | mit<br>Wert<br>(hex) | Bedeutung                            |
|---------------------------------|----------------------|--------------------------------------|
| MOTORCHARACTERISTIC_EEPROM_LOAD | 17<br>(hex)          | Load motor configuration from EEPROM |
| GET_CPU_TEMP                    | 18<br>(hex)          | Read temperature from CPU            |
| GET_MOTOR_SUPPLY_VOLTAGE        | 19<br>(hex)          | Read voltage from CPU                |
| GO_POSITION_RELATIVE            | 20<br>(hex)          | Drive to relative position           |



### 3.6.8.2. Values for par 1 of function SET\_MOTORCHARACTERISTIC

| Parameter<br>(DAPI_STEPPER_MOTORCHAR<br>_PAR_ ...) | Value<br>(dec) | Description                                   |
|--|----------------|---|
| STEPMODE   | 1              | Stepmode (Full-, 1/2-, 1/4-, 1/8-, 1/16-step) |
| GOFREQUENCY  | 2              | Speed [Full-step / s] - related to full-step  |
| STARTFREQUENCY                                     | 3              | Startfrequency [Full-step / s]                |
| STOPFREQUENCY                                      | 4              | Stopfrequency [Full-step / s]                 |
| MAXFREQUENCY                                       | 5              | Maximum frequency [Full-step / s]             |
| ACCELERATIONSLOPE                                  | 6              | Acceleration slope [Full-step / ms]           |
| DECELERATIONSLOPE                                  | 7              | Deceleration slope [Full-step / 10ms]         |
| PHASECURRENT                                       | 8              | Phase current [mA]                            |
| HOLDPHASECURRENT                                   | 9              | Phase current for motor hold [mA]             |
| HOLDTIME   | 10             | Time in that the hold goes to motorstop [ms]  |
| STATUSLEDMODE                                      | 11             | Mode of the Status-LED                        |
| INVERT_ENDSW1                                      | 12             | Invert endswitch1                             |
| INVERT_ENDSW2                                      | 13             | Invert endswitch2                             |

| Parameter<br>(DAPI_STEPPER_MOTORCHAR<br>_PAR_ ...) | Value<br>(dec) | Description  |
|--|----------------|--|
| INVERT_REFSW1                                      | 14             | Invert referenceswitch1                                |
| INVERT_REFSW2                                      | 15             | Invert referenceswitch2                                |
| INVERT_DIRECTION                                   | 16             | Invert all direction details                           |
| ENDSWITCH_STOPMODE                                 | 17             | Setting of the stop behaviour<br>(0=Fullstop / 1=Stop) |
| GOREFERENCEFREQUENCY_TOE<br>NDSWITCH               | 18             | Frequency before endswitch [Full-<br>step / s]         |
| GOREFERENCEFREQUENCY_AFT<br>ERENDSWITCH            | 19             | Frequency after endswitch [Full-<br>step / s]          |
| GOREFERENCEFREQUENCY_TOO<br>FFSET                  | 20             | Frequency to optional offset<br>[Full-step / s]        |

### 3.6.8.3. Values for par 1 of function GO\_REFSWITCH

| Parameter<br>(DAPI_STEPPER_GO_<br>REFSWITCH_PAR_ ...) | Value<br>(dec) | Description                                 |
|---|----------------|---|
| REF1  | 1              | Drive to reference switch 1                 |
| REF2  | 2              | Drive to reference switch 2                 |
| REF_LEFT  | 4              | Drive to the left edge of reference switch  |
| REF_RIGHT   | 8              | Drive to the right edge of reference switch |
| REF_GO_POSITIVE                                       | 16             | Drive motor right way                       |
| REF_GO_NEGATIVE                                       | 32             | Drive motor left way                        |
| SET_POS_0   | 64             | Set position of motor to 0                  |

### 3.6.8.4. Examples

#### The command

```
DapiStepperCommand(handle, 1, DAPI_STEPPER_CMD_GO_POSITION,  
3200, 0, 0, 0);
```

will be send in a 8 byte CAN package.

#### Structure of the CAN package:

| CAN-Byte | Type             | Value | Byte |
|----------|------------------|-------|------|
| 1        | COMMAND          | 4     | 04   |
| 2        | PAR1 (Bit 0-7)   | 3200  | 80   |
| 3        | PAR1 (Bit 8-15)  |       | 0C   |
| 4        | PAR1 (Bit 16-23) |       | 00   |
| 5        | PAR1 (Bit 24-31) |       | 00   |
| 6        | PAR2 (Bit 0-7)   | 0     | 00   |
| 7        | PAR2 (Bit 8-15)  |       | 00   |
| 8        | PAR3 (Bit 0-7)   | 0     | 00   |

### 3.6.9. PWM outputs

Structure of a 8 Byte length CAN package.

| CAN-Data-Byte | Content                 |
|---------------|-------------------------|
| 1             | PWM channel 1 (Bit 0-7) |
| 2             | PWM channel 2 (Bit 0-7) |
| 3             | PWM channel 3 (Bit 0-7) |
| 4             | PWM channel 4 (Bit 0-7) |
| 5             | PWM channel 5 (Bit 0-7) |
| 6             | PWM channel 6 (Bit 0-7) |
| 7             | PWM channel 7 (Bit 0-7) |
| 8             | PWM channel 8 (Bit 0-7) |

## Ethernet (TCP) protocol



IV

## **4. Ethernet (TCP) protocol**

### **4.1. The TCP protocol**

Communication takes place using the TCP/IP protocol on the ethernet interface. A network connection between the module and a PC is assumed.

#### **4.1.1. IP-adress**

The IP-address is modifiable by using the integrated web interface. This is necessary, if the IP-address is already in use.

#### **4.1.2. Netmask**

If the IP-address is outside of the IP-address range and/or the IP-address range is subdivided differently, then the netmask must be modified using the integrated webserver.

#### **4.1.3. Port**

The TCP-transfer to send and receive is realized using the "default Port 9912". Should this be a problem (e.g. due to a firewall), than the port may be modified (à manual RO-Series).

## 4.2. TCP request/response string

At the beginning, the request string will be explained in detail with a request string example. Afterwards, the request string will be explained with a response to the request example.

### 4.2.1. Request string structure

16-bit address mode

#### Example for a write command with 16-bit address mode:

The following example writes the data 0f hex to the address 0012 hex (the request string is build up of 11 characters). The JOB-ID is 01.

| Character no. | Description                         | Value                      | Hexadecimal |
|---------------|-------------------------------------|----------------------------|-------------|
| 1             | Packet_ID_0                         |                            | 63          |
| 2             | Packet_ID_1                         |                            | 9a          |
| 3             | TCP_CMD_RO_1                        |                            | 01          |
| 4             | JOB_ID                              |                            | 01          |
| 5             | Length of request string (bit 8-15) | 11 dec.                    | 00          |
| 6             | Length of request string (bit 0-7)  | (amount of all characters) | 0b          |
| 7             | COMMAND                             | ASCII "W"                  | 57          |
| 8             | WIDTH (data width)                  | ASCII "B"                  | 42          |
| 9             | ADDRESS bit 8-15                    | 0012 hex                   | 00          |
| 10            | ADDRESS bit 0-7                     |                            | 12          |
| 11            | DATA (= 0f hex)                     | 0f hex                     | 0f          |



Here you can find the 32-bit address mode:

**32-bit address mode**

## 32-bit address mode

### Example for a write command with 32-bit address mode:

The following example writes the data 0f hex to the address 80010012 hex (the request string is build up of 14 characters). The JOB-ID is 01.

| Character no. | Description                         | Value                      | Hexadecimal |
|---------------|-------------------------------------|----------------------------|-------------|
| 1             | Packet_ID_0                         |                            | 63          |
| 2             | Packet_ID_1                         |                            | 9a          |
| 3             | TCP_CMD_RO_1                        |                            | 01          |
| 4             | JOB_ID                              |                            | 01          |
| 5             | Length of request string (bit 8-15) | 14 dec.                    | 00          |
| 6             | Length of request string (bit 0-7)  | (amount of all characters) | 0e          |
| 7             | Extended Address Mode               | ASCII "X"                  | 58          |
| 8             | COMMAND                             | ASCII "W"                  | 57          |
| 9             | WIDTH (data width)                  | ASCII "B"                  | 42          |
| 10            | ADDRESS Bit 24-31                   | 80010012 hex               | 80          |
| 11            | ADDRESS Bit 16-23                   |                            | 01          |
| 12            | ADDRESS Bit 8-15                    |                            | 00          |
| 13            | ADDRESS Bit 0-7                     |                            | 12          |
| 14            | DATA (= 0f hex)                     | 0f hex                     | 0f          |

Here you can find the 16-bit address mode:

## 16-bit address mode

#### 4.2.1.1. JOB-ID

The JOB-ID denotes the actual request string. Two adjacent request strings may not have the same JOB-ID.

##### **HINT:**

After a successful transmission, the request routine should increase the JOB-ID by "1". That way, the numbers from "0" to "255" and then "0" again are transferred successively.

#### 4.2.1.2. Length of the request string

The length of the request string is build up of 2 bytes and indicates how much bytes the request string is build of. The most significant bits are transmitted first. In the example, the length is 11 bytes.

#### 4.2.1.3. Extended address mode

**The Extended Address Mode** (32-bit addresses) will be used by products which have a 32-Bit address register. The **X** in the request string shows, if the 32-Bit mode is used.

#### 4.2.1.4. COMMAND

With COMMAND, the character "W" is sent to write to registers while "R" is used to read from registers.

#### 4.2.1.5. WIDTH

In dependency of the used "COMMAND" and the data "WIDTH", the sent data is different in width.

| COMMAND | WIDTH | Width of data (bits) | Amount bytes |
|---------|-------|----------------------|--------------|
| W       | B     | 8                    | 1            |
| W       | W     | 16                   | 2            |
| W       | L     | 32                   | 4            |
| W       | X     | 64                   | 8            |
| R       |       | 0                    | 0            |

The COMMAND = "R" has no influence to the WIDTH-character of the request string. It controls the data length of the response string.

#### **4.2.1.6. ADDRESS**

The registers are 16 bit or 32 bit large and are represented by 2 or 4 hexadecimal characters. During the transmission, the most significant bits are sent first.

The transmission for 16 bit addresses begins with bits 8-15, then bits 0-7 follows.

The transmission for 32 bit addresses begins with bits 24-31, 16-23, 8-15, then bits 0-7 follows.

#### **4.2.1.7. DATA**

The data field only exists during a write command. It is left out for a read command.

### 4.2.2. Response string structure

The response string is in hexadecimal and begins with 2 bytes. They are labeled as PACKET\_ID\_0 (hex 63) and PACKET\_ID\_1 (hex 9a) and followed by TCP\_ANSWER\_RO\_1 (hex 81).

Now comes an error code and successive hexadecimal characters. In case the request string is a read command, read-out data will be appended at the end of the string.

OK-response to the example's request string from chapter 4

| Character no. | Description                          | Value | Hexadecimal |
|---------------|--------------------------------------|-------|-------------|
| 1             | DEDITEC_PACKET_ID_0                  |       | 63          |
| 2             | DEDITEC_PACKET_ID_1                  |       | 9a          |
| 3             | DEDITEC_TCP_ANSWER_RO_1              |       | 81          |
| 4             | OK (OK = 0x00)                       |       | 00          |
| 5             | JOB-ID                               |       | 01          |
| 6             | Length of response string (bit 8-15) | 7     | 00          |
| 7             | Length of response string (bit 0-7)  |       | 07          |

#### 4.2.2.1. Error code

The error code indicates with 0 (hex) = ok, that the requester's data were successfully received. Any other error code indicates an error.

#### **4.2.2.2. JOB-ID**

The received JOB-ID from the requester is sent back. Two consecutive request strings may not have the same JOB-ID.

#### **4.2.2.3. Length of the response string**

The length of the response string consists of 2 bytes and indicates the byte-length of the response string. The most significant bits are sent first. In the example, the length is 7 bytes.

#### **4.2.2.4. DATA**

The data field only exists in case of a reply to a read command. A reply to a write command has no data field.

## Serial protocol





## **5. Serial protocol**

### **5.1. Speed and interface parameters**

The serial interface is configured with 115200 bauds, 8 data bits, 1 Stop bit and no parity (8,n,1).

### **5.2. Structure of the „sending string“**

The string begins with a <SOH> character (“Start of Header”, 01 hex) and ends with a <CR> character (“Carriage-Return”, 0D hex).

Before transmission, hexadecimal data is ASCII-encoded. Not encoded are begin and end characters as well as COMMAND and WIDTH. Accessing for example the address 0012 hex, results in transmittling from left to right the 4 ASCII characters “0012” consecutively. ASCII encoded hexadecimal values may only have the numbers from 0..9 and the letters A..F.

### Write command example:

The following example writes the data word 0F hex to the address 0012 hex (the sending string is build up of 16 ASCII-characters)

| Byte no.     | Description  | ASCII-character | Hexadecimal    |
|--------------|--|-----------------|----------------|
| 1            | Start Of Header<br>(marks begin of string)                             | <SOH>           | 01             |
| 2, 3         | MODULE-NO (=34 hex)  | 34              | 33, 34         |
| 4, 5         | JOB-ID (= 12 hex)  | 12              | 31, 32         |
| 6            | COMMAND  | W               | 57             |
| 7            | WIDTH (data width)   | B               | 42             |
| 8, 9, 10, 11 | ADDRESS (= 0012 hex)   | 0012            | 30, 30, 31, 32 |
| 12, 13       | DATA (= 0F hex)  | 0F              | 30, 46         |
| 14, 15       | CHECKSUM ( = 29D hex,<br>the lower 8 bits will be<br>only transmitted) | 9D              | 39, 44         |
| 16           | END (Carriage Return)  | <CR>            | 0D             |

#### 5.2.1. Start character

The <SOH> character indicates the begin of a data transfer.

### **5.2.2. MODUL-NO**

The MODULE-NO indicates the selected module (the module numbers can be modified using the DIP switches on the modules itself).

### **5.2.3. JOB-ID**

The JOB-ID indicates the actual sending string. Two consecutive sending strings may not have the same JOB-ID.

HINT:

After a successful data transfer, the sending routine should increase the JOB-ID by "1". That way, the successive numbers "0", then "1" .. "255" and again "0" are sent.

### **5.2.4. COMMAND**

With COMMAND, the characters "W" or "R" indicates writing to or reading from registers.

### 5.2.5. WIDTH

In dependence of the used "COMMAND" and data width, a different data length is transferred.

| COMMAND | WIDTH | Data width (bits) | Amount of ASCII characters |
|---------|-------|-------------------|----------------------------|
| W       | B     | 8                 | 2                          |
| W       | W     | 16                | 4                          |
| W       | L     | 32                | 8                          |
| W       | X     | 64                | 16                         |
| R       |       | 0                 | 0                          |

The COMMAND = "R" has no influence on the transferred data length. It controls the data length of the receive string

### 5.2.6. ADDRESS

The registers are 16 bits wide and are represented with 4 hexadecimal values. As the transfer of every hexadecimal value is done with an ASCII character, 4 ASCII characters are needed.

### 5.2.7. DATA

When a write command is used, the data field present. A read command does not have any data field.

#### 5.2.8. CHECKSUM

CHECKSUM is a simple checksum. It is a hexadecimal summation of all characters (including <SOH>) until the checksum itself. The receiver compares the summation with the transferred checksum. If both are equal, the transfer is seen as error-free.

##### In hexadecimal:

$01 + 33 + 34 + 31 + 32 + 57 + 42 + 30 + 30 + 31 + 32 + 30 + 46 = 29D$

→ CHECKSUM = 9D hex (2 is the overflow and is therefore left out).

#### 5.2.9. END

The end of the sending string is indicated with a <CR> (Carriage Return, 0D hex).

### 5.3. Structure of the „response string“

The response string begins with a character (“O”, “D” oder “E”) followed by ASCII characters and ends with a <CR> (Carriage Return, 0d hex).

#### 5.3.1. Response-string „O“ (OK)

This means the response is “ok” and informs the sender that the write command is successfully executed.

##### “O” - format

| Byte no. | Description            |
|----------|------------------------|
| 1        | ASCII-“O” for OK       |
| 2        | JOB-ID (higher byte)   |
| 3        | JOB-ID (lower byte )   |
| 4        | CHECKSUM (higher byte) |
| 5        | CHECKSUM (lower byte)  |
| 6        | END (Carriage Return)  |

### OK-response of the sending string example of chapter 3.2

| Byte no. | Description           | ASCII-character | Hexadecimal |
|----------|-----------------------|-----------------|-------------|
| 1        | OK                    | O               | 4F          |
| 2, 3     | JOB-ID (= 12 hex)     | 12              | 31, 32      |
| 4, 5     | CHECKSUM ( = B2 hex)  | B2              | 42, 32      |
| 6        | END (Carriage Return) | <CR>            | 0D          |

### 5.3.2. Response-string „D“ (DATA)

A response-string beginning with a “D” indicates response data, resulting from a read command.

#### “D” - format

| Byte no. | Description                                  |
|----------|--|
| 1        | “D” for DATA<br>(response to a read command) |
| 2        | JOB-ID (higher byte)                         |
| 3        | JOB-ID (lower byte)                          |
| 4,5 ...  | n-data                                       |
| 4 + n    | CHECKSUM (higher byte)                       |
| 5 + n    | CHECKSUM (lower byte)                        |
| 6 + n    | END (Carriage Return)                        |

With n = 2, 4, 8 or 16 characters (depending on the WIDTH of the sending string)



Command + WIDTH of the sending string and the response data

| Read-request |       | Response-data        |                             |
|--------------|-------|----------------------|-----------------------------|
| Command      | WIDTH | Data width<br>(bits) | Amount ASCII-<br>character. |
| R            | B     | 8                    | 2                           |
| R            | W     | 16                   | 4                           |
| R            | L     | 32                   | 8                           |
| R            | X     | 64                   | 16                          |

### 5.3.3. Start-character with response „E“ (ERROR)

The last sent sending string was incorrectly received.

#### “E” - format

| Byte no. | Description           |
|----------|-----------------------|
| 1        | “E” for error         |
| 2        | Error code            |
| 3        | END (Carriage Return) |

#### Error code

| Hexadecimal | Description  |
|-------------|--|
| 31          | Incorrect command; an invalid COMMAND was used                 |
| 32          | Invalid data length; the length of the sending-string is false |
| 33          | Checksum error   |

# Appendix



## 6. Appendix

### 6.1. Revisions

|          |  |
|----------|--|
| Rev 3.01 | DEDITEC Design Update 2022   |
| Rev 3.00 | DEDITEC Design Update 2021   |
| Rev 2.12 | Added "Auto Reactivate" and "Secure" timeout commands                    |
| Rev 2.11 | CAN-RX mode for PWM outputs added  |
| Rev 2.10 | Extended address mod for Ethernet protocol added                         |
| Rev 2.00 | Design change<br>Merging of all protocol manuals and register assignment |

## 6.2. Copyrights and trademarks

Linux is a registered trademark of Linus Torvalds.

USB is a registered trademark of USB Implementers Forum Inc.

LabVIEW is a registered trademark of National Instruments.

Intel is a registered trademark of Intel Corporation.

AMD is a registered trademark of Advanced Micro Devices, Inc.

ProfiLab is a registered trademark of ABACOM Ingenieurbüro GbR.

ispVM System is a registered trademark of Lattice Semiconductor Corporation.

Windows, Visual-C/C++, -C#, -Basic, -Basic.NET and Visual-Studio are registered trademarks of Microsoft Corporation.

Delphi is a registered trademark of Borland Software Corporation.

Java is a registered trademark of Oracle Corporation.