

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFx Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

# CANopen user's manual for SICK Identification Sensors CLV6xx, RFH/RFU6xx, Lector6xx

| Date       | Version | Author        | Changes  |
|------------|---------|---------------|--|
| 2009 12 01 | 1.0     | Aschenbrenner | initial revision   |
| 2009 12 06 | 1.1     | Aschenbrenner | minor corrections  |
| 2009 12 08 | 1.2     | Aschenbrenner | corrections Jerry Finley included  |
| 2010 06 08 | 1.3     | Adler         | Additional infos for Result Output via PDO   |
| 2011 01 27 | 1.4     | Thomas        | Inclusion for ID <sup>pro</sup> devices RFH6xx and Lector  |
| 2011 11 30 | 1.5     | Klümper       | Correction of subindex for CAN inputs and outputs  |
| 2021 02 28 | 1.7     | Aschenbrenner | Generalized for different identification sensors.<br>New EDS-files for each type of identification sensor.<br>(CLV – Lector – RFH – RFU )<br>The object directories are identical for all different types of identification sensors! |

## Related Documents:

CiA Draft Standard 301  
 CANopen application layer and communication profile  
 ( <http://www.can-cia.de/> )

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

## **Contents**

|            |  |           |
|------------|--|-----------|
| <b>1</b>   | <b>CAN NETWORKING WITH SICK IDENTIFICATION SENSORS .....</b>   | <b>3</b>  |
| <b>2</b>   | <b>CANOPEN OBJECT DIRECTORY.....</b>                           | <b>4</b>  |
| <b>3</b>   | <b>ACCESS TO SENSORS READ RESULT .....</b>                     | <b>8</b>  |
| <b>3.1</b> | <b>READ RESULT ACCESS BY SDO UPLOAD .....</b>                  | <b>9</b>  |
| <b>3.2</b> | <b>READ RESULT DATA TRANSFER BY PDO .....</b>                  | <b>11</b> |
| <b>3.3</b> | <b>CONFIGURING TPDOS USING CANOPEN MECHANISMS.....</b>         | <b>12</b> |
| <b>4</b>   | <b>I/O COMMUNICATION .....</b>                                 | <b>14</b> |
| <b>4.1</b> | <b>CANOPEN INPUTS (SLAVE DEVICE OUTPUTS) (OBJECT 0X6000)..</b> | <b>14</b> |
| <b>4.2</b> | <b>CANOPEN OUTPUTS (SLAVE DEVICE INPUTS) (OBJECT 0X6200)..</b> | <b>15</b> |
| <b>4.3</b> | <b>PDO MAPPING FOR DIGITAL I/O .....</b>                       | <b>15</b> |
| <b>5</b>   | <b>SOPAS COMMANDS VIA CANOPEN.....</b>                         | <b>21</b> |
| <b>6</b>   | <b>READING DIAGNOSIS VIA CANOPEN (ONLY CLV6XX).....</b>        | <b>21</b> |
| <b>7</b>   | <b>CANOPEN HEARTBEAT OBJECTS.....</b>                          | <b>22</b> |
| <b>8</b>   | <b>DEVICE SPECIFIC HEARTBEAT TELEGRAMS.....</b>                | <b>22</b> |
| <b>9</b>   | <b>THE EMERGENCY OBJECT .....</b>                              | <b>22</b> |
| <b>9.1</b> | <b>ASSIGNING THE EMERGENCY OBJECT ID.....</b>                  | <b>22</b> |
| <b>9.2</b> | <b>EMERGENCY OBJECT CONTENT .....</b>                          | <b>23</b> |

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFx Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

## 1 CAN networking with SICK identification sensors

Most of the SICK identification sensors, Barcode scanners CLV, RFID readers RFH/RFU and Identification cameras Lector, have a CAN interface. It can be used in two different operation modes: SICK-Network or CANopen.

Using SICK-Network, the CAN interface is initialized for interconnection to different SICK Autodent products. This networking bases on the CANopen protocol, but it is not intended to have other products besides SICK identification sensors and controllers in the network.

It is not the subject of this document to describe the functions of the CAN SICK-Network.

If you select CANopen for The CAN Mode, the SICK identification sesnors will behave as a CANopen slave device which may work in any CANopen network together with any other CANopen device.

The screenshot shows a configuration window titled "CAN". It contains the following settings:

- Mode:** A dropdown menu set to "CANopen".
- Use Device-ID as Node-ID:** A checked checkbox.
- Device ID:** A text input field containing the value "6".
- Baudrate:** A dropdown menu set to "250 kBit/sec (max. 250m)".

If you want to run several devices in a CANopen network you must ensure that each device uses the same baudrate and that each device has its specific node ID.

For SICK ident sensors we use the device ID also as its CANopen Node ID.

**Note:** If you are using an external CMC600 parameter cloning module mounted in the CLVs Connection box, you can select the node ID by the address switches and you also can select the CAN baudrate and the CANopen mode by setting the mode switches to a specific position. (See CMC600 operating instructions)

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

## 2 CANopen Object Directory

Each CANopen slave device has a CANopen object directory (OBD). It describes all the data objects of the device which can be accessed (read or write) by data transfer on the CAN bus.

In case of SICK ident devices, we have a very huge object directory. This is because the CANopen protocol is also used for the CAN SICK-network.

Most of the entries should not be used by customers.

CLV and RFH devices really have those objects in CANopen mode, but objects users should not use are not listed in the electronic data sheet (EDS-file).

Lector and RFU use different object directories for sick-network or CANopen mode.

In both cases there are the same objects a CANopen user may use.

In deed there is the same OBD for any device. Independent if it is CLV, Lector, RFU or RFH.

SICK identification sensors have a huge amount of different parameters. Don't think all of them may be accessed via CANopen by just accessing the object directory. You normally need to setup you device using SOPAS ET. The CANopen interface primarily was made to communicate the devices IOs and their read results.

Additionally it is possible to send SOPAS commands and to receive commands responses. This way you nevertheless can indirectly access a devices parametes and of course, it is possible to call SOPAS methods. This is the the way for writing data to tags when using a RFID device.

This is the OBD used for each of the SICK ID senors:

### Communication segment

| Index | Subindex | Name                           | type           | value   |
|-------|----------|--------------------------------|----------------|---------|
| 1000h | 00h      | Device type                    | unsigned 32    | 0x30191 |
| 1001h | 00h      | Error register                 | unsigned 8     |         |
| 1002h | 00h      | Manufacturer status register   | unsigned 32    |         |
| 1003h | 00h .. n | Predefined error field         | unsigned 32    |         |
| 1005h | 00h      | COP-ID Sync message            | unsigned 32    |         |
| 1008h | 00h      | Manufacturers Device Name      | Visible string |         |
| 1009h | 00h      | Manufacturers Hardware Version | Visible string |         |
| 100Ah | 00h      | Manufacturers Software Version | Visible string |         |
| 1010h | 01       | p301_store_para                | unsigned 32    |         |

|             |                  |                     |                              |  |
|-------------|------------------|---------------------|------------------------------|--|
| <b>SICK</b> | Abteilung:       | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |  |
|             | EA-Nr./Int.-Nr.: |                     | <b>CANopen User's Manual</b> |  |
|             | Projektleiter:   |                     |                              |  |
|             | Bearbeiter:      |                     |                              |  |

|              |                 |                               |             |              |
|--------------|-----------------|-------------------------------|-------------|--------------|
| 1011h        | 01              | p301_restore_para             | unsigned 32 |              |
| 1014h        | 00              | COB-ID emergency object       | unsigned 32 |              |
| 1015h        | 00              | Inhibit time emergency        | unsigned 16 |              |
| 1017h        | 00              | Producer Heartbeat time       | unsigned 16 |              |
| 1018h        |                 | Identity Object               |             |              |
|              | 00              | number of entries             | unsigned 8  |              |
|              | 01              | Vendor ID                     | unsigned 32 | 0x0056       |
|              | 02              | Product Code                  | unsigned 32 |              |
|              | 03              | Revision number               | unsigned 32 |              |
|              | 04              | Serial Number                 | unsigned 32 |              |
| 1200h        |                 | Server SDO Parameter 1        |             |              |
|              | 00              | Number of entries             | unsigned 8  |              |
|              | 01              | COB-ID Client → Server        | unsigned 32 |              |
|              | 02              | COB-ID Server → Client        | unsigned 32 |              |
| <b>Index</b> | <b>Subindex</b> | <b>Name</b>                   | <b>type</b> | <b>value</b> |
| 1400h        | 00h .. 02h      | Receive PDO comm. param. 1    | u8 / u32    |              |
| 1401h        | 00h .. 02h      | Receive PDO comm. param. 2    | u8 / u32    |              |
| 1402h        | 00h .. 02h      | Receive PDO comm. param. 3    | u8 / u32    |              |
| 1403h        | 00h .. 02h      | Receive PDO comm. param. 4    | u8 / u32    |              |
| 1600h        | 00h .. 08h      | Receive PDO mapping param. 1  | u8 / u32    |              |
| 1601h        | 00h .. 08h      | Receive PDO mapping param. 2  | u8 / u32    |              |
| 1602h        | 00h .. 08h      | Receive PDO mapping param. 3  | u8 / u32    |              |
| 1603h        | 00h .. 08h      | Receive PDO mapping param. 4  | u8 / u32    |              |
| 1800h        | 00h .. 05h      | Transmit PDO comm. param. 1   | u8 / u32    |              |
| 1801h        | 00h .. 05h      | Transmit PDO comm. param. 2   | u8 / u32    |              |
| 1802h        | 00h .. 05h      | Transmit PDO comm. param. 3   | u8 / u32    |              |
| 1803h        | 00h .. 05h      | Transmit PDO comm. param. 4   | u8 / u32    |              |
| 1805h        | 00h .. 05h      | Transmit PDO comm. param. 6   | u8 / u32    |              |
| 1806h        | 00h .. 05h      | Transmit PDO comm. param. 7   | u8 / u32    |              |
| 1807h        | 00h .. 05h      | Transmit PDO comm. param. 8   | u8 / u32    |              |
| 1808h        | 00h .. 05h      | Transmit PDO comm. param. 9   | u8 / u32    |              |
| 1809h        | 00h .. 05h      | Transmit PDO comm. param. 10  | u8 / u32    |              |
| 180Ah        | 00h .. 05h      | Transmit PDO comm. param. 11  | u8 / u32    |              |
| 180Bh        | 00h .. 05h      | Transmit PDO comm. param. 12  | u8 / u32    |              |
| 1A00h        | 00h .. 08h      | Transmit PDO mapping param. 1 | u8 / u32    |              |
| 1A01h        | 00h .. 08h      | Transmit PDO mapping param. 2 | u8 / u32    |              |
| 1A02h        | 00h .. 08h      | Transmit PDO mapping param. 3 | u8 / u32    |              |
| 1A03h        | 00h .. 08h      | Transmit PDO mapping param. 4 | u8 / u32    |              |
| 1A05h        | 00h .. 08h      | Transmit PDO mapping param. 6 | u8 / u32    |              |
| 1A06h        | 00h .. 08h      | Transmit PDO mapping param. 7 | u8 / u32    |              |

|             |                  |                     |                              |  |
|-------------|------------------|---------------------|------------------------------|--|
| <b>SICK</b> | Abteilung:       | <b>GBC08 – BU81</b> | <b>CLV RfX Lector</b>        |  |
|             | EA-Nr./Int.-Nr.: |                     | <b>CANopen User´s Manual</b> |  |
|             | Projektleiter:   |                     |                              |  |
|             | Bearbeiter:      |                     |                              |  |

|       |            |                                |          |  |
|-------|------------|--------------------------------|----------|--|
| 1A07h | 00h .. 08h | Transmit PDO mapping param. 8  | u8 / u32 |  |
| 1A08h | 00h .. 08h | Transmit PDO mapping param. 9  | u8 / u32 |  |
| 1A09h | 00h .. 08h | Transmit PDO mapping param. 10 | u8 / u32 |  |
| 1A0Ah | 00h .. 08h | Transmit PDO mapping param. 11 | u8 / u32 |  |
| 1A0Bh | 00h .. 08h | Transmit PDO mapping param. 12 | u8 / u32 |  |

### Manufacturer Segment (Used CLV6xx barcode data)

| Index | Subindex   | Name  | type           | access |
|-------|------------|---|----------------|--------|
| 2000h |            | Read result   | unsigned 32    |        |
| 2000h | 00h        | Number of entries (value = 4)   | unsigned 8     | RO     |
| 2000h | 01h        | counter (each read result)  | unsigned 8     | RO     |
| 2000h | 02h        | length of datastring  | unsigned 16    | RO     |
| 2000h | 03h        | data valid / free data on write   | unsigned 8     | R/WW   |
| 2000h | 04h        | Selected output format datastring                                       | Visible string | RO     |
|       |            |   |                |        |
| 2001h |            | Successive single characters of read result                             | Visible string |        |
| 2001h | 00h        | Number of entries   | unsigned 8     | RO     |
| 2001h | 01d .. 50d | One character on each subindex  | unsigned 8     | RO     |
|       |            |   |                |        |
| 2002h | 00h        | Counter for successive read results (used to see if result has changed) | unsigned 8     | RO     |
|       |            |   |                |        |
| 2010h |            | reading diagnosis datastring (same as on serial AUX interface)          | unsigned 32    |        |
| 2010h | 00h        | Number of entries (value = 4)   | unsigned 8     | RO     |
| 2010h | 01h        | counter (each successive output )                                       | unsigned 8     | RO     |
| 2010h | 02h        | length of datastring  | unsigned 16    | RO     |
| 2010h | 03h        | data valid / free data on write   | unsigned 8     | R/WW   |
| 2010h | 04h        | reading diagnosis datastring  | Visible string | RO     |
|       |            |   |                |        |
|       |            |   |                |        |
| Index | Subindex   | Name  | type           | access |
| 2020h |            | command response datastring   | unsigned 32    |        |
| 2020h | 00h        | Number of entries (value = 4)   | unsigned 8     | RO     |
| 2020h | 01h        | counter (each successive output )                                       | unsigned 8     | RO     |
| 2020h | 02h        | length of datastring  | unsigned 16    | RO     |
| 2020h | 03h        | data valid / free data on write   | unsigned 8     | R/WW   |
| 2020h | 04h        | command response datastring   | Visible string | RO     |
| 2200h |            | command datastring  | domain (500)   | WO     |
|       |            |   |                |        |
| 2300h |            | structure 'data available' (usually mapped to TPD= )                    |                |        |
| 2300h | 00h        | Number of entries (value = 7)   | unsigned 8     | Const  |
| 2300h | 01h        | Length of data to be uploaded   | unsigned 16    | RWR    |
| 2300h | 02h        | Type of data  | unsigned 8     | RWR    |

|             |   |                     |                              |  |
|-------------|---|---------------------|------------------------------|--|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RfX Lector</b>        |  |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |  |

|        |     |  |             |       |
|--------|-----|--|-------------|-------|
| 2300h  | 03h | counter  | unsigned 8  | RWR   |
| 2300h  | 04h | not used   | unsigned 8  | RWR   |
| 2300h  | 05h | Node ID (of source = Tx device)  | unsigned 8  | RWR   |
| 2300h  | 06h | dummy  | unsigned 8  | RWR   |
| 2300h  | 07h | dummy  | unsigned 8  | RWR   |
|        |     |  |             |       |
| 3000h  | 00h | Enable bits for device sending data strings via CAN:<br>Bit0: Enable Read Result Datatstring (2000h / 4)<br>Bit1: Enable Diagnosis Data (2010h / 4)<br>Bit2: Enable Command Response (2020h / 4) | unsigned 8  | RWW   |
|        |     |  |             |       |
| 0x3010 |     | Timeouts / ms for dynamic queued OBD entries.  |             |       |
| 0x3010 | 00h | Number of entries (value = 3)  | unsigned 8  | Const |
| 0x3010 | 01h | timeout read result string   | unsigned 16 | RW    |
| 0x3010 | 02h | timeout diagnosis string   | unsigned 16 | RW    |
| 0x3010 | 03h | timeout command response string  | unsigned 16 | RW    |

### Device Profile Segment ( Digital I/O)

| Index | Subindex | Name  | type       | access |
|-------|----------|---|------------|--------|
| 6000h |          | CANopen inputs ( = Slave device output )                  |            |        |
| 6000h | 00h      | Number of entries (value = 2)                             | unsigned 8 | RO     |
| 6000h | 01h      | Digital input byte 0                                      | unsigned 8 | RO     |
| 6000h | 02h      | Digital input byte 1                                      | unsigned 8 | RO     |
| 6200h |          | CANopen outputs ( = slave device input )                  |            |        |
| 6200h | 00h      | Number of entries (value = 2)                             | unsigned 8 | RO     |
| 6200h | 01h      | Digital output byte 0                                     | unsigned 8 | RWW    |
| 6200h | 02h      | Digital output byte 1                                     | unsigned 8 | RWW    |
| 6208h |          | Enable for CANopen outputs<br>( for Slave device inputs ) |            |        |
| 6208h | 00h      | Number of entries (value = 2)                             | unsigned 8 | RO     |
| 6208h | 01h      | Digital output byte 0                                     | unsigned 8 | RW     |
| 6208h | 02h      | Digital output byte 1                                     | unsigned 8 | RW     |

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV Rfx Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

### 3 Access to sensors read result

The CAN interface provides data transfer of either output format #1 or output format #2 like all the other data interfaces of the sensor. Output formats are data strings that the sensor constructs after the end of each reading cycle. You can describe their format in the section 'Data Processing' \ 'Output Format' of the SOPAS engineering tool.

Just like for all other data interfaces of the sensor you can select for the CAN interface, which one of both possible data formats will be communicated.

There are two different methods to transfer read results using the CANopen protocol:

The first of them is uploading the data string to the PLC by initializing a SDO domain upload. This same implementation was already used for the CLV4xx barcode scanners. Below you will find a description how to process this.

The second method is to map the successive characters of the read result to one or several PDOs that triggered each time a reading cycle was finished.

The selected data format will be put to the object directory. The reading result is located at 0x2000/04. It may contain up to 500 characters (max.). At 0x02000/01 we have a counter, which is incremented with each successive read result. It should be used to see if new data has been sent. (Remember: There might be two successive read results with the same content!) At 2000h/02 there is an entry that has the current length of the current result data string.

The object 0x2001 also contains the readresult. This is a data array including the first 50 characters of the read result data frame. Each subindex has a single character. If the length is less than 50 bytes the last array elements will be set to 00h. If it is more than 50 then the last part of this string is lost for this object.



|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

### 3.1 Read result Access by SDO upload

CANopen slave devices cannot initialize SDO data transfer themselves. They are always passive. It is a CANopen client's (e.g. a PLC's) job to initialize a SDO upload or download.

Therefor the slave devices (reader devices) send specific PDO data for signaling that new data is available at object 0x2000. It is called 'data available PDO', described below. A client which receives this PDO should start an SDO upload procedure to get the read result data domain. After the domain upload was finished, the client should signal to its server device that the read result data can be released. It can do this by writing 0x00h to the data valid byte, which is 2000h/03h for the read result domain.

The readers 'Data Available PDO' or 'Sync-PDO' has a structure as described below :

| <b>Data available PDO:</b> |              |   |
|----------------------------|--------------|---|
| Identifier:                | 11 bit       | 0x180 + Node id<br>(like predefined connection set for TPD01) |
| 1st mapped object:         | unsigned 16: | length of the read result data domain                         |
| 2nd mapped object:         | unsigned 8:  | type identifier   |
| 3rd mapped object:         | unsigned 8:  | counter, incremented for each read result                     |
| 4th mapped object:         | unsigned 8:  | not used  |
| 5th mapped object          | unsigned 8:  | PDO source device Node ID                                     |

The type identifiers on the 2nd position of the mapped object, tells us which kind of data is available and at which position in the object directory we can find it:

|      |                              |          |
|------|------------------------------|----------|
| 0x01 | Read result data string      | 0x2000/4 |
| 0x04 | Command response data string | 0x2020/4 |
| 0x05 | Diagnosis data String        | 0x2010/4 |

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RfX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

|      |   |          |
|------|---|----------|
| 0x81 | error: timeout 'Read result data string' reached      | 0x2000/4 |
| 0x84 | error: timeout 'Command response data string' reached | 0x2020/4 |
| 0x85 | error: timeout 'Diagnosis data String' reached        | 0x2010/4 |

Example for data traffic on the CAN line

Scenario: a slave device (NodeID = 3) has read result data. Length = 10 bytes

| CAN-Identifier              | CAN Object type<br>/ direction  | CAN Data          | comment  |
|-----------------------------|---|-------------------|--|
| <b>Synchronisations-PDO</b> |   |                   |  |
| 0x183                       | PDO von CLV03   | 0A 00 01 00 03 03 | Synchronisations-Pdo<br>Type                    01h: Read Result Datastring<br>Length                0Ah: 10 Zeichen<br>Counter:              00h<br>Digout:                01h<br>Knotennummer        03h |
| <b>SDO upload</b>           |   |                   |  |
| 0x603                       | Handshake for SDO upload of domain object 0x2000/04   |                   |  |
| 0x583                       |   |                   |  |
| <b>SDO Release String</b>   |   |                   |  |
| 0x603                       | Handshake for SDO download<br>Write <b>0</b> to <b>2000h / 3</b> : This will release the current read result datastring.<br>If there is a next read result in the queue, then immediately a next Sync PDO with ID 0x183 will be started.<br>If the queue is empty. The next sync PDO will start after there is a next read result available and entered to the OBD. |                   |  |
| 0x583                       |   |                   |  |

When the server sends its 'Sync-PDO', it also starts a timeout to check if the client is starting the upload procedure. If the timer runs out, result data that was attached to the object directory will automatically be released and a second PDO with an error type identifier (bit 7 = 1) will be sent.

Mode to send ReadResult
by SDO
Timeout ReadResult
2000
Automatic Release
☐

The timeout time (in ms) can be entered using SOPAS ET.

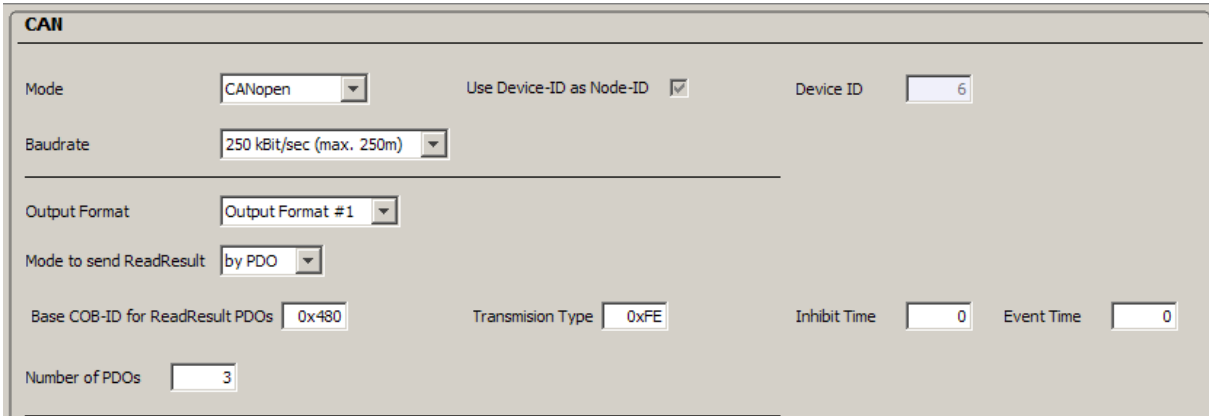
There is also a checkbox 'Automatic Release'. If this is active, data will be released automatically after the SDO upload process finished successfully. A second access the the same read result then won't be possible.

If the client tries to upload read result data while there is no valid read result is in the OBD, an abort message 0x08000022 will be generated.

This means: 'data cannot be transferred because of present device state'

## 3.2 Read result data transfer by PDO

– arranged by SOPAS ET GUI



It is also possible to transfer the read result data string using PDOs and to define the nservicelevelnumber of PDOs to be sent. The CAN object IDs used may be defined using SOPAS ET. In this case they are successive number defined by the 'BASE COB-ID' wich is the lowest number, used for the first of the PDOs.

Of course, it is possible to define the COB-IDs by writing to the TPDO communication parameters in the OBD of the device.

There may be a single PDO with up to 7 characters of the string or a set of PDOs where each PDO has a set of up to 7 successive characters of the string.

Up to 49 characters of the read result data string can be sent. Each PDO carries 7 bytes of the string and an additional counter data byte, which is incremented for each successive read result. You need to check this counter byte to ensure if you have consistent readresult data within the set of different PDOs.

AS the PDOs are sent with successive COB IDs in the example you will get 3 PDOs with Identifiers 0x480, 0x481 and 0x482.

| COBID | data Byte1 = Counter | byte 2..8(ReadResult)       | Datybates ASCII |
|-------|----------------------|-----------------------------|-----------------|
| 480h  | 05h                  | 52h 65h 61h 64h 52h 65h 73h | - R e a d R e s |
| 481h  | 05h                  | 75h 6Ch 74h 00h 00h 00h 00h | - u l t - - - - |
| 482h  | 05h                  | 00h 00h 00h 00h 00h 00h 00h | - - - - - - -   |

| COBID | data Byte1 = Counter | byte 2..8(ReadResult)       | Datybates ASCII |
|-------|----------------------|-----------------------------|-----------------|
| 480h  | 06h                  | 31h 32h 33h 34h 35h 36h 37h | - 1 2 3 4 5 6 7 |
| 481h  | 06h                  | 38h 39h 30h 41h 42h 43h 44h | - 8 9 0 A B C D |
| 482h  | 06h                  | 45h 46h 47h 48h 00h 00h 00h | - E F G H - - - |

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

In the tables above you can see two successive read results. The first readresult consists of a 10 character datastring "Readresult". The second has 18 charcaters: "1234567890ABCDEFGH"

If the transmission type ID is 0xFE (see CANopen spec: asynchronous transmission), a PDO will be sent on change of data. This is at the end of each reading cycle, directly after the selected output format (#1 or #2) was built. If you want to have a cyclic transmission of PDOs you can enter the event time != 0.

It is also possible to map these 50 result characets (CANopen Object 0x2001 Subindex 1-50) to any Transmit PDO (see chapter 4.3.2 Viewing PDO setings within Sopas ET). The parameter "Mode to send ReadResult" has to be set to "by PDO".

### 3.3 Configuring TPDOs using CANopen mechanisms

Probably you want to arrange the TPDO configurations of the device for transmitting a read result using your own CANopen configuration tool. This method not only configures the identification devics but also your CANopen controller, which must be prepared for receiving.

Consider: Unfortunately, there are some restrictions:

**TPDO1** has a fixed mapping for the Sync PDO shown above. You can use the identifier of the predefined connection set (0x180+nodeID) or your own assignement. Don't change the object mapping.

When you store your CANopen setting by writing to object 0x1010, the settings will be internally transferred to to SOPAS setup and you should be able to see it in the SPOAS ET GUI.

**TPDO2, TPDO3 and TPDO4** may be universally used.

Also the settings will be shown in the SOPAS GUI after storing. (Write obj. 0x1010)

**TPDO5..TPDO12** are internally used if you define by SOPAS GUI to transfer the read result by PDO.

The mapping is fixed. You cannot change it by writing to their TPD mapping objects 0x1A05 .. 0x1A0B.

For each of them the first mapped object is 0x2002/00. This is the round robin counter, which oncrements on each next read result. You should use it to see of each of the partial read results fits to the same reading cycle and so to see if your read result consisting from several different PDO tranfers is consistent.

The next seven mappings are successive characters as a part of the complete read result data string:

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

|        |                  |                        |
|--------|------------------|------------------------|
| TPDO6  | Char1 .. Char7   | 0x2001/01 .. 0x2001/07 |
| TPDO7  | Char8 .. Char14  | 0x2001/08 .. 0x2001/0E |
| TPDO8  | Char15 .. Char21 | 0x2001/0F .. 0x2001/16 |
| TPDO9  | Char22 .. Char28 | 0x2001/17 .. 0x2001/1D |
| TPDO10 | Char29 .. Char35 | 0x2001/1E .. 0x2001/25 |
| TPDO11 | Char36 .. Char42 | 0x2001/26 .. 0x2001/2C |
| TPDO12 | Char43 .. Char49 | 0x2001/2D .. 0x2001/34 |

You can assign a COB ID for TPD06 by writing to 0x1805. It will be stored when writing to OBJ0x1010.

You even can assign individual COB IDs for TPDO7 to TPDO12. However, this is dangerous!

Its work temporarily. When restarting the device or when storing the configuration other COB IDs will be assigned in a way that each is a successive number beginning at the value used for TPDO6.

If you want to use TPDO6 .. TPDO12 you should take the predefined mappings and you should assign successive COB ID beginning with any COBID used for TPDO6

Remember: For your CANopen system you have to be very careful not to use any identifier more than once !

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFx Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User´s Manual</b> |

## 4 I/O Communication

The sensors fieldbus I/O data of course may be distributed via CAN. A CANopen master device has direct access via SDO communication to the objects 0x6000 (CANopen inputs = Slave device output) and 0x6200 (CANopen outputs = slave device input). There is also an object 0x6208 with enable bits, which define, which output bits (= sensor input bits) will be handled.

### 4.1 CANopen inputs (slave device outputs) (object 0x6000)

| Bit           | object   | assignment | name                            | comment                     |
|---------------|----------|------------|---------------------------------|-----------------------------|
| Byte 0, Bit 0 | 0x6000/1 | fixed      | Device Ready                    |                             |
| Byte 0, Bit 1 | 0x6000/1 | fixed      | System Ready                    | Not for CANopen             |
| Byte 0, Bit 2 | 0x6000/1 | fixed      | Good Read                       |                             |
| Byte 0, Bit 3 | 0x6000/1 | fixed      | No Read                         |                             |
| Byte 0, Bit 4 | 0x6000/1 | fixed      | Status External Output 1        | Physical Output 1 of CDF600 |
| Byte 0, Bit 5 | 0x6000/1 | fixed      | Status External Output 2        | Physical Output 2 of CDF600 |
| Byte 0, Bit 6 | 0x6000/1 | fixed      | Status Output 1 (Result 1)      |                             |
| Byte 0, Bit 7 | 0x6000/1 | fixed      | Status Output 2 (Result 2)      |                             |
| Byte 1, Bit 0 | 0x6000/2 | fixed      | External Input 1                | Physical Input 1 of CDF600  |
| Byte 1, Bit 1 | 0x6000/2 | fixed      | External Input 2                | Physical Input 2 of CDF600  |
| Byte 1, Bit 2 | 0x6000/2 | fixed      | Input 1 (Sensor 1)              |                             |
| Byte 1, Bit 3 | 0x6000/2 | fixed      | Input 2 (Sensor 2)              |                             |
| Byte 1, Bit 4 | 0x6000/2 | soft       | Defined by sensor configuration | Not yet implemented         |
| Byte 1, Bit 5 | 0x6000/2 | soft       | Defined by sensor configuration | Not yet implemented         |
| Byte 1, Bit 6 | 0x6000/2 | soft       | Defined by sensor configuration | Not yet implemented         |
| Byte 1, Bit 7 | 0x6000/2 | soft       | Defined by sensor configuration | Not yet implemented         |

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

## 4.2 CANopen outputs (slave device inputs) (object 0x6200)

| Bit           | object   | assignment | name                        | comment                     |
|---------------|----------|------------|-----------------------------|-----------------------------|
| Byte 0, Bit 0 | 0x6200/1 | fixed      | Trigger                     |                             |
| Byte 0, Bit 1 | 0x6200/1 | fixed      | Sensor-Idle                 |                             |
| Byte 0, Bit 2 | 0x6200/1 | fixed      | TeachIn1                    |                             |
| Byte 0, Bit 3 | 0x6200/1 | fixed      | TeachIn2                    |                             |
| Byte 0, Bit 4 | 0x6200/1 | fixed      | External Output_1           | Physical Output 1 of CDF600 |
| Byte 0, Bit 5 | 0x6200/1 | fixed      | External Output_2           | Physical Output 2 of CDF600 |
| Byte 0, Bit 6 | 0x6200/1 | fixed      | Digital Output_1 (Result_1) |                             |
| Byte 0, Bit 7 | 0x6200/1 | fixed      | Digital Output_2 (Result_2) |                             |
| Byte 1, Bit 0 | 0x6200/2 | soft       | PLC_Out_08                  |                             |
| Byte 1, Bit 1 | 0x6200/2 | soft       | PLC_Out_09                  |                             |
| Byte 1, Bit 2 | 0x6200/2 | soft       | PLC_Out_10                  |                             |
| Byte 1, Bit 3 | 0x6200/2 | soft       | PLC_Out_11                  |                             |
| Byte 1, Bit 4 | 0x6200/2 | fixed      | Distance_Config_0           | LSB                         |
| Byte 1, Bit 5 | 0x6200/2 | fixed      | Distance_Config_1           |                             |
| Byte 1, Bit 6 | 0x6200/2 | fixed      | Distance_Config_2           |                             |
| Byte 1, Bit 7 | 0x6200/2 | fixed      | Distance_Config_3           | MSB                         |

## 4.3 PDO mapping for digital I/O

The inputs and outputs that are listed in the object directory can be mapped to PDOs.

The device has 4 receive PDOs (RPDO1 .. RPDO4) and 4 transmit PDOs (TPDO1 .. TPDO4) which can be individually mapped. Digital outputs (sensor inputs, 0x6200) can be mapped to RPDOs while digital inputs (0x6000) can be mapped to TPDOs. There are different methods how to do this:

### 4.3.1 Mapping by writing to OBD (standard CANopen method)

A CANopen user can enter a setup for TPDO 1..4 and for RPDO 1..4 like it is common for CANopen slave devices. By writing to the Objects 0x1400 .. 0x1403 he can set the RPDO communication parameters. 0x1600 .. 0x1603 is used for RPDO mapping parameters, 0x1800 .. 0x1803 for TPDO communication parameters and 0x1A00 .. 0x1A03 for TPDO mapping parameters

These parameters can be stored permanently (together with the whole device parameter set) by writing value 0x65766173 (= 'save') to the Object 0x1010/01.(see CANopen spec )

They are part of the sensors parameter set and so they are also be stored in an external parameter cloning device. (CMC600)

**Note:** If you want to use the 'Data Available PDO' (Sync-PDO) which was described above, you should not use TPDO1 for your application.

### 4.3.2 Viewing PDO settings within Sopas ET

On the CANopen pages of the SOPAS ET GUI for the identification sensors the configurations for TPDO1 to TPDO4 and for RPDO1 to RPADO 4 are shown.

It even is possible to change the configuration with this tool, but we don't recommend to do this. It is difficult to see and to track the synchronization of this parameters to the device. Furthermore it is favorable and more usual to take the users CANopen configuration tool to do this.

If Sopas ET is connected to the device while there is CANopen configuration access to it then the shown CANopen setting shown in the GUI are synchronized to the device settings each time a 'write parameters' – operation is done by writing the storing code to object =x1010/01.

#### 4.3.2.1 TPDOs

This is the way the TPDO configuration is shown in SPOAS ET:

| CANopen Transmit PDOs 1 .. 4 |            |            |            |            |
|------------------------------|------------|------------|------------|------------|
|                              | TPDO1      | TPDO2      | TPDO3      | TPDO4      |
| Predef. conn.-               | Yes        | No         | No         | Yes        |
| COB-ID                       | 0x80000000 | 0x00000184 | 0x00000185 | 0x80000000 |
| Transm. Type                 | 0xFE       | 0xFE       | 0xFE       | 0xFE       |
| Inhibit Time                 | 0          | 0          | 0          | 0          |
| Event Time                   | 0          | 0          | 0          | 0          |
| Num of map, o                | 7          | 2          | 2          | 0          |
| Transmit PDOs                | Map Obj. 1 | 0x23000110 | 0x60000108 | 0x60000208 |
|                              | Map Obj. 2 | 0x23000208 | 0x60000208 | 0x60000108 |
|                              | Map Obj. 3 | 0x23000308 | 0x00000000 | 0x00000000 |
|                              | Map Obj. 4 | 0x23000408 | 0x00000000 | 0x00000000 |
|                              | Map Obj. 5 | 0x23000508 | 0x00000000 | 0x00000000 |
|                              | Map Obj. 6 | 0x23000608 | 0x00000000 | 0x00000000 |
|                              | Map Obj. 7 | 0x23000708 | 0x00000000 | 0x00000000 |
|                              | Map Obj. 8 | 0x00000000 | 0x00000000 | 0x00000000 |

For CLV and RFH devices the used object identifier are not shown while the predefined connection set is used. The background of the values therefore is grayed out in this case. Lector and RFU device show the real Identifiers even in this case.



|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RfX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

For the transmission types the coding define by the CANopen specification is shown.  
0xFE in the example below means that there is asynchronous transmission of the TPDOs.  
(Probably mostly used)

The inhibit time is a minimum delay before sending a PDO a second time.  
It prevents the system from being blocked by a very frequent PDOs. Mostly this parameter is not used and therefore set to 0. (Unit is 100us)

The event time is the cycle time in ms when cyclically transmitting PDOs.

To describe mapped objects 32 bit data values are used.

This is the data format:

|                |                  |  |     |
|----------------|------------------|--|-----|
| MSB            |                  |  | LSB |
| Index (16 bit) | Subindex (8 bit) | Object length - num of <b>bits</b> (8 bit) |     |

Example: '0x60000108' describes the 'Digital input byte 0':

Index = 0x6000, Subindex = 0x01, Number of Bits = 0x08

(See CiA DS 301 9.6.3 Object 1600h - 17ffh)

In the shown SOPAS configuration example above there are three TPDOs mapped.

TPDO1 has the mapping for the sync-PDO ('Data available PDO') which is automatically entered if you select 'SDO' data transfer for the read result. The COB-ID is default and selects the predefined connection set identifier. It is 0x180 + Node ID for the first TPDO.

If the NodeID of the device is 6, then the data available PDO will be sent with COB-ID 0x186.

TPDO2 of the example above maps the two digital input data bytes 0x6000/01 and 0x6000/02 and assigns the Identifier 0x184. The event time is 0, so the PDO will be sent each time when any of the digital input bits changes.

TPDO3 of the example also maps the two digital input data bytes but in different order. It assigns the Identifier 0x185. The event time is 500, so the PDO will be sent every 500 ms (It additionally is sent immediately if any of the digital input bits changes).

Remark: using identifiers 0x184 and 0x185 as shown in this example might be dangerous, because they might conflict to the predefined connection set identifiers of node No 4 and node No 5.

#### 4.3.2.2 RPDOs

| CANopen Receive PDOs 1 .. 4 |            |            |            |            |
|-----------------------------|------------|------------|------------|------------|
|                             | RPDO1      | RPDO2      | RPDO3      | RPDO4      |
| Predef. conn.               | No         | No         | Yes        | Yes        |
| COB-ID                      | 0x00000220 | 0x00000230 | 0x80000000 | 0x80000000 |
| Transm. Type                | 0xFE       | 0xFE       | 0xFE       | 0xFE       |
| Num of map. o               | 1          | 1          | 0          | 0          |
| Map Obj. 1                  | 0x6200108  | 0x62000208 | 0x00000000 | 0x00000000 |
| Map Obj. 2                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| Map Obj. 3                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| Map Obj. 4                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| Map Obj. 5                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| Map Obj. 6                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| Map Obj. 7                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| Map Obj. 8                  | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

In the RPDO configuration example above, we have two receive PDOs for the device. Each of them receives a single Byte.

RPDO1 receives the low byte of the digital output object 0x6200 / 01. (Length = 08 bits)

The assigned COB-ID is 0x220. A CAN object with Identifier 0x220 can be used to trigger the device. Bit 0 of 0x6200 / 01 is the trigger bit.

| Bit           | object   | assignment | name              |
|---------------|----------|------------|-------------------|
| Byte 0, Bit 0 | 0x6200/0 | fixed      | Trigger           |
| Byte 0, Bit 1 | 0x6200/0 | fixed      | Sensor-Idle       |
| Byte 0, Bit 2 | 0x6200/0 | fixed      | TeachIn1          |
| Byte 0, Bit 3 | 0x6200/0 | fixed      | TeachIn2          |
| Byte 0, Bit 4 | 0x6200/0 | fixed      | External Output_1 |
| Byte 0, Bit 5 | 0x6200/0 | fixed      | External Output_2 |

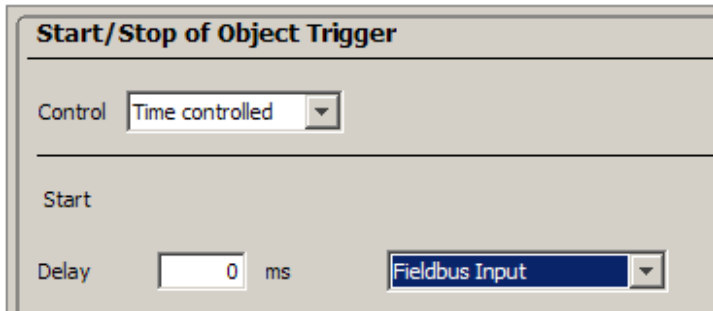
You need some other settings in the SOPAS setup of the device, to really enable triggering by this PDO:

The digital input mask (which is the digital output mask from the PLCs point of view) must be set:

|                     |      |
|---------------------|------|
| Mask for Dig. Input | 0x11 |
|---------------------|------|

In this example bit 1 and bit 5 are enabled. So the digital output of the PLC is enabled to access the the external output (Bit5) and the trigger bit.

In addition, to enable this trigger functionality you must select the 'Fielbus Input' for the trigger configuration in SOPAS.



**Start/Stop of Object Trigger**

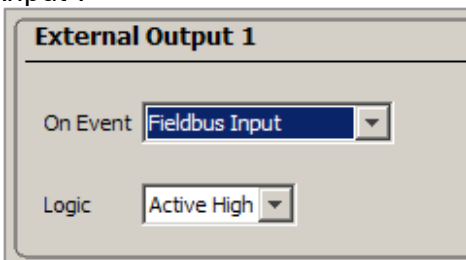
Control: Time controlled

Start

Delay: 0 ms

Fieldbus Input

For setting the External Output (Output available on CDB600 or CDM400) of the device directly by PDO the function off the digital output bit must be configured as “Fieldbus Input”.



**External Output 1**

On Event: Fieldbus Input

Logic: Active High


Digital output byte 1 of the device (0x6200/01) is mapped within the second RPDO.

It has its own identifier and so another source device may access those output bits.

|               |          |       |                   |     |
|---------------|----------|-------|-------------------|-----|
| Byte 1, Bit 0 | 0x6200/1 |       | PLC_Out_08        |     |
| Byte 1, Bit 1 | 0x6200/1 | soft  | PLC_Out_09        |     |
| Byte 1, Bit 2 | 0x6200/1 | soft  | PLC_Out_10        |     |
| Byte 1, Bit 3 | 0x6200/1 | soft  | PLC_Out_11        |     |
| Byte 1, Bit 4 | 0x6200/1 | fixed | Distance_Config_0 | LSB |
| Byte 1, Bit 5 | 0x6200/1 | fixed | Distance_Config_1 |     |
| Byte 1, Bit 6 | 0x6200/1 | fixed | Distance_Config_2 |     |
| Byte 1, Bit 7 | 0x6200/1 | fixed | Distance_Config_3 | MSB |

Bits 7 .. 4 of this byte can be used to change the distance configuraton of the CLV6xx. As shown above, you need to select some other switching parameters, to get the distance configuration changes by the fieldbus master device:

The mask for the fielbus input (of the device) must be enabled.



Mask for Dig. Input: 0xF011

Furthermore the distance configuration settings must be assigned to the fieldbus device. (If using a barcode scanner that has dynamic focus functionality as an option)

To get the menu below you must login to the barcode scanner at “Service Level” in SOPAS and select a specific menu path:

Menu: ReadingConfiguration - activate 'dynamic reading configuration'

The screenshot shows the 'Codelabel Properties' dialog box. It contains several settings: Scan frequency (900 Hz), Inverse Code (unchecked), Quietzone Ratio (Auto), Codelabel Distance (175 mm), Codelabel Quality (Standard), Minimum Reading Angle (0), and Maximum Reading Angle (100). The 'Dyn. Reading Config.' checkbox is highlighted with a red circle.

Select 'more' !

This screenshot shows the 'Codelabel Properties' dialog box with 'Dyn. Reading Config.' checked. The 'more...' button is highlighted with a red circle. Below the checkbox, there is a table with 8 rows, each labeled 'Cfq. 1' through 'Cfq. 8' and a 'Code, Qual.' column, all set to 'Standard'.

|        | Code, Qual. |
|--------|-------------|
| Cfq. 1 | Standard    |
| Cfq. 2 | Standard    |
| Cfq. 3 | Standard    |
| Cfq. 4 | Standard    |
| Cfq. 5 | Standard    |
| Cfq. 6 | Standard    |
| Cfq. 7 | Standard    |
| Cfq. 8 | Standard    |

The 'Dynamic Control mode' must be set to 'Fieldbus'

The distance configuration bits 0..3 entered by our RPDO will be handled as index value ( 0..7) for the resulting distance configuration.

The screenshot shows two sections: 'General Settings' and 'Assignment table'. In 'General Settings', 'Dynamic control mode' is set to 'Fieldbus' (highlighted with a red circle) and 'Behavior' is set to 'Immediate'. The 'Assignment table' section shows 'Assignment table length' set to 8 and a row of 8 index selectors (1 to 8).

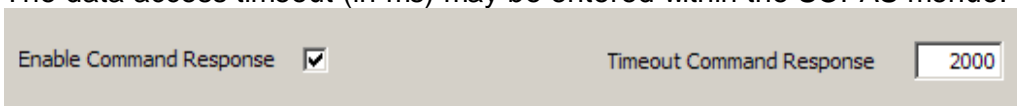
|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

## 5 Sopas commands via CANopen

A CANopen client device can send SOPAS commands to the device, just as commands can be sent via any interface (Serial port, Ethernet, USB). In case of CANopen the client (the CANopen PLC) has to start a SDO download and write the command to the object directory of the addressed server device. You can find the command input object at index 0x2200/00 in the sensors OBD. The sensor will interpret each command, after the SDO download sequence has finished. It will put its command response in object 0x2020 and start a sync PDO ('data available PDO') as described above for sending read result data strings. In case of a command response the type identifier within the PDO is 0x04 (see table in chapter 3.1)

The procedure to get the command response is the same as loading read results. Except the OBD-entry to be accessed is 0x2020/04 and data release must be done by write access (data = 0) to 0x2020/03

The data access timeout (in ms) may be entered within the SOPAS menu.



Enable Command Response ☒      Timeout Command Response

## 6 Reading diagnosis via CANopen (only CLV6xx)

Reading diagnosis data strings which normally are sent on the AUX interface, can also be directed to the CANopen interface.



Enable Diagnosis Output ☒      Timeout Diag. Output

Getting the diagnosis data string is the same procedure as for read results and command response datastrings. The 'data available PDO' will use a type identifier 0x05 (see table in chapter 3.1) . The diagnosis data strings can be uploaded from Object 0x2010/4

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

## 7 CANopen heartbeat objects

Heartbeat Time / ms

3000

CANopen heartbeat objects (CAN objects with identifier 0x700 + NodeID) can be enabled by setting the heartbeat time to value != 0.

Of course heartbeating can as well be enabled by writing to object 0x1017 = Producer Heartbeat Time, as for any other CANopen slave device.

## 8 Device specific heartbeat telegrams

Enable Heartbeat ☒

Heartbeat Interval  s

Restart Interval on Sending ☒

This checkmark can be used to enable sending of heartbeat telegrams via CANopen. Heartbeat telegrams are specific datastrings which are defined in the data format section of the SOPAS menu. They will be sent in the same way as read result data strings, in case read results are missing (for a while) because there is no identification object in the system.

It is just the same behavior as if a heartbeat telegram is sent on a serial interface instead of a read result telegram.

Our CANopen subsystem handles heartbeat telegrams in the same way as read result datastrings. It's the same Object 0x2000/04. Even object 0x2002/00 is incremented when a heartbeat data string was generated

## 9 The Emergency Object

### 9.1 Assigning the emergency object ID

COB-ID Emergency Obj.

0x0

Emcy Inhibit Time

0

The emergency object ID can be assigned using the SOPAS engineering tool.

You can also set the Emergency Object ID by writing to Object 0x1014 using your CANopen configuration tool.

## 9.2 Emergency Object Content

Several software instances of the sensor device can detect irregular states within the sensor. There is a shared instance, called errorhandler, which collects information about status events which are detected all over the device.

Each warning can be identified by a 32 bit value which is divided in several bit groups.

Bit 31..24: Errorlevel:

|             |      |
|-------------|------|
| Debug Error | 0x01 |
| Info        | 0x02 |
| Warning     | 0x03 |
| Error       | 0x04 |
| Fatal Error | 0x05 |

Bit 17 .. 8: Subsystem where error was detected

|                        |      |
|------------------------|------|
| Genral errors          | 0x00 |
| Error for test purpose | 0x01 |
| CAN-Network            | 0x02 |
| Network general        | 0x03 |
| Network monitor        | 0x04 |
| External devices       | 0x07 |

Bit 7 .. 0: Error numbers within a subsystem

The errorhandler notifies each event to the CANopen system of the device, which enters entries into the object directory. This way sending of emergency objects is triggered.

All error events are put into the manufacturer status register 0x1002 formatted in the way that was is shown above. If there is an error reset event on an error type that was allready entered to object 0x012, then the entry will be deleted.

NOTE: If there was already a second error event put into the fifo, when the reset event happens, the error cannot be deleted within the fifo.

Unfortunately the CANopen error sytem cannot handle the 32 bit error states of our device.

So the errors are mapped to the CANopen error system as described below:

Entries to the error register (object 0x1001):

0x81 → generic error bit and manufacturer specific error bit will be set

### Data of the emergency object

Emergency error Code (Bytes 0 and 1 of the Emergency Object)

|             |   |                     |                              |
|-------------|---|---------------------|------------------------------|
| <b>SICK</b> | Abteilung:  | <b>GBC08 – BU81</b> | <b>CLV RFX Lector</b>        |
|             | EA-Nr./Int.-Nr.:<br>Projektleiter:<br>Bearbeiter: |                     | <b>CANopen User's Manual</b> |

Highbyte = 0xFF (→ DS301: 'Device specific error')

Lowbyte = Errorcode not an explicit entry, no subsystem information

Error register entry (Byte 2 of the emergency object)

0x81 for device specific errors.

Manufacturer specific error field (Byte 3 and 4 of the emergency object)

(= Byte 3 and 2 of the predefined error field)

Byte 3: error generating subsystem

Byte 4: error level