HYDAC ELECTRONIC

Functional Safety
PL d
SIL 2

CANopen
safety easy to use

Protocol Description CANopen Safety

HAT 1000 / HAT 3000

Single Turn Angle Sensor

(Translation of original instructions)

Part No.: 669985 / Revised: 2020-11-05 E

Table of Contents .

Preface

This manual provides you, as user of our product, with key information on the operation and maintenance of the equipment.

It will acquaint you with the product and assist you in obtaining maximum benefit in the applications for which it is designed.

Keep the manual in the vicinity of the instrument for immediate reference.
Please note that the data on the software technology provided in this manual refers to that available at the time of publication.

If you discover errors while reading the documentation or have additional suggestions or tips, please contact us at:

HYDAC ELECTRONIC GMBH
Technical Documentation
Hauptstrasse 27
66128 Saarbruecken
-Germany-
Phone: +49(0)6897 / 509-01
Fax: +49(0)6897 / 509-1726
Email: electronic@hydac.com

We look forward to receiving your input.

“Putting experience into practice”

1 Introduction

The angle sensor series HAT CANopen Safety meet the CANopen standards according to the following profiles and standards:

[1] CiA DS301, Version: 4.2.0 (21 February 2011)
CANopen application layer and communication profile

[2] CiA DS302, Version: 4.1 (02 February 2009)
Additional application layer functions - Part 2: Network management

[3] CiA DS303, Version: 1.4 (14th August 2006)
Additional specification - Part 2: Representation of SI units and prefix

[4] CiA DS304
Version: 1.0.1, (1st January 2005)
Framework for safety-relevant communication

[5] CiA DS305, Version: 2.2 (26th August 2008)
Layer setting services (LSS) and protocols

[6] CiA DS406, Version 3.2 (18th December 2006)
Device profile for encoders

This manual describes the functions supported by the HAT CANopen Safety. A basic knowledge of CAN and CANopen and CANopen Safety is assumed. The exact function is described in the a.m. *Draft Standards*. Since both specifications are issued in English, the features described in this manual are identified using the English description from the specification and are shown in *italics*, for clarity.

2 Functions of the HAT CANopen Safety

- **Determination of the turning angle (Single Turn) between the shaft and the housing (absoute)**
- Transmission of the actual angle value as a **PDO or SRDO** for the following events:
  - **Synchronously** in relation to received SYNC objects (PDO only)
  - **Asynchronously**, cyclically, within the range from 1 millisecond to >1 minute

3 Transmission rates

The HAT CANopen Safety supports the following transmission rates (Baud rates):

- 1000 kbit/s
- 800 kbit/s
- 500 kbit/s
- 250 kbit/s
- 125 kbit/s
- 50 kbit/s
- 20 kbit/s
- 10 kbit/s

The timing complies with DS301, *Bit rates and timing*.

The transmission rate used is stored in a non-volatile memory.
When supplied, it is set to **500 kbit/s** and can be changed via the CANbus (see *Object Dictionary Index 2002h*).

4 CAN Frames

The HAT CANopen Safety supports the 11 bit base frames with 11 bit identifiers, as well as 29 bit frames with 29 bit identifiers, required in the specification.

5 Node ID

To operate the HAT CANopen Safety in a CANopen network a unique *Node ID* must be set within the network.

The set *Node ID* is stored in a non-volatile memory and can be adjusted via the CAN bus (see *Object Dictionary Index 2001h*) or LSS[5].
When supplied, the address **1** is set.

6 Transmission services

6.1 Service Data Object (SDO)

With CANopen, all the device's data (setting parameters and measured data) are filed in an *Object Dictionary* under a specified *Index*. Various entries of the *Object Dictionary* are sub-divided still further using a *Subindex*. Using the *SDOs,* other network nodes can read from or write to the *Object Dictionary* of the HAT CANopen Safety.
The HAT CANopen Safety takes on the role of a *Server*, and the device intending to read or write the data, takes on the role of a *Client*.

To transmit data the HAT CANopen Safety must have a *Receive-SDO* serving for the reception of data and a *Transmit-SDO* which serves for sending data. Sequence of the data transmission:

Reading from Object Dictionary:

1. A device (*Client*) sends the Receive-SDO of the HAT CANopen Safety (*Server*). This *SDO* contains an identification to say that the *Object Dictionary* is to be read, as well as the *Index* and *Subindex*.
2. The HAT CANopen Safety (*Server*) sends its Transmit-SDO. The *Index* and the *Subindex*, as well as the read data are also included herein.

Writing to the Object Dictionary:

1. A device (*Client*) sends the Receive-SDO of the HAT CANopen Safety (*Server*). This SDO contains an identification, to say that the Object Dictionary is to be written to, as well as the required index, subindex and the data to be entered.
2. The HAT CANopen Safety (*Server*) sends its Transmit-SDO. This SDO also contains the index and the subindex, as well as an identification to say the Object Dictionary has been written to.

If an error should occur, e.g. the specified *Index* does not exist, or an attempt is made to write to a *read only* entry, or the data is not within the valid range, then the Transmit-SDO receives an appropriate *Abort SDO Transfe*r identification and a corresponding *Abort Code* (see [1]).

The particular *COB-ID* of the *SDO* corresponds to the *Pre-defined Connection Set* defined in the DS301 *a*nd cannot be altered.

COB IDs for Service Data Objects

SDO	***COB ID***
Receive – SDO	600h+Node-ID
Transmit – SDO	580h+Node-ID

6.2 Process Data Object (PDO)

Data transmission using *SDOs* is indeed very flexible, but has some disadvantages when transmitting measured values or actuating variables: only one piece of data can be read, the data must first be requested with an *SDO* and because the relevant *Index* and *Subindex* is also transmitted, what is known as the overhead increases further.

For this reason CANopen Safety defines what is known as *Process Data Objects*. These contain only the necessary useful data. There are two types of *PDO*:

1. *Transmit PDOs*
   These enable the measuring instrument to send its measured values.
2. *Receive PDOs*
   These enable the actuating variables to be transmitted to an actuator or a controller.

What is known as the *PDO Mapping* stipulates which data is now in a *PDO*. This *PDO Mapping* is stored in the *Object Dictionary* (see *Object Dictionary, Index 1A00h*).

The *PDO Transmission Type* stipulates with which ID and for which event a *PDO* is transmitted. These settings are also stored in the *Object Dictionary* (see *Object Dictionary, Index 1800h*).

Events which result in a PDO being sent:

1. Receipt of a *SYNC* object (synchronous transmission).
2. Expiry of an adjustable cycle time within the range from 1 milliseconds to >1 minute (cyclical transmission).

The HAT CANopen Safety implements one *Transmit-PDOs*, which includes the actual sensor or process values.

The DS406 stipulates the transmission of the actual measured value as a 16 bit value or an 32 bit value, as the default setting.
The HAT CANopen Safety only implements the transmission as a 32 bit value.

6.3 Synchronisation Object (SYNC)

SYNC objects are used to implement a synchronous data transmission. A *SYNC* object is in fact a CAN message with a defined identifier, without data. CANopen differentiates between *SYNC Producers* and *SYNC Consumers. SYNC Producers* are devices on the bus which send a *SYNC* at adjustable time intervals. *SYNC Consumers* are devices which react to receiving a *SYNC*. In a CANopen network several *SYNC* objects can exist. The individual *SYNC* objects are differentiated by means of the *SYNC ID* which corresponds to the CAN identifier used. The *SYNC ID* used is stored in the *Object Dictionary*.

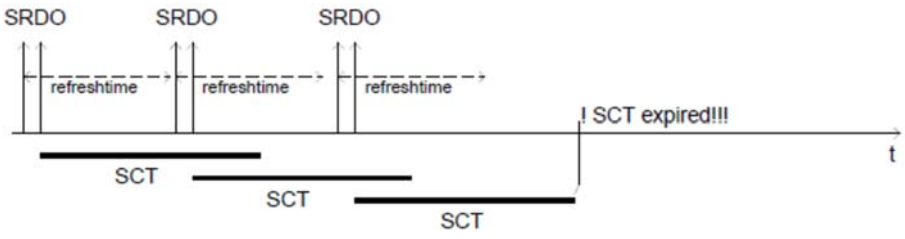
The HAT CANopen Safety provides the functionality of a *SYNC Consumer*. If the *PDO Transmission Type* is set appropriately, a *PDO* is sent on receipt of a *SYNC*. The *SYNC-ID* is pre-set to 80h and can be changed in the *Object Dictionary* (see *Object Dictionary, Index 1005h*). In *PDO Transmission Type* the number of received SYNC objects which result in a PDO being sent, can be set.

6.4 Safety-relevant Data Object (SRDO)

The data transmission via what is known as *Safety-relevant Data Objects* ensures a safe transfer of the measured values. SRDOs can be used or deactivated in order to send or receive data.

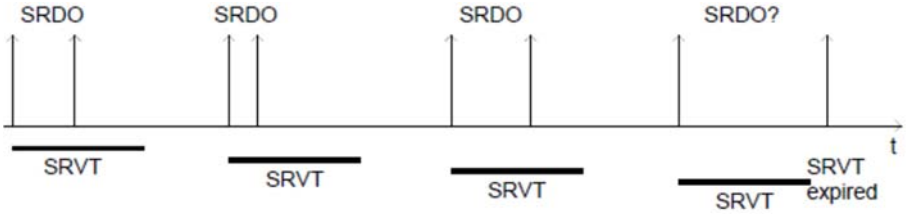
SRDOs consist of two autonomous CAN frames. The first frame has data in normal display, the second CAN frame has the data in inverted display. It enables the receiver to verify the data.

The time between two SRDOs is called *Refresh Time* or *Safeguard cycle time (SCT)* and serves for timeout monitoring between two SRDOs.



Source: CiA Draft Standard 304, CANopen Framework for safety-relevant communication Version: 1.0.1, January 2005, CAN in Automation (CiA), page 9

The time between the two frames is called *Safety-relevant object validation time (SRVT)* and enables the receiver to have a timeout recognition within one SRDO.



Source: CiA Draft Standard 304, CANopen Framework for safety-relevant communication,Version: 1.0.1, January 2005, CAN in Automation (CiA),page 10

What is known as the *SRDO Mapping* stipulates which data is now in a SR*DO*. This mapping is stored in the *Object Dictionary*.

The firmware supports one Transmit-SRDO and no Receive-SRDOs.

The following items should be observed during configuration:

- The modification of the communication parameters is possible only in *Pre-operational status.*
- To modify the COB-IDs, the corresponding *SRDO* must first be deactivated (*Information Direction* is 0).
- The SCT on the receiver node must be chosen accordingly higher than on the sender node in order to avoid synchronisation problems due to software runtimes (±2.8 ms).
- The SRVT is to be chosen in such a way that an interruption of the SRDOs on the CANbus can not lead to a failure at the receiver due to higher valued messages.
- The *SCT* and *SRVT* are as well highly depending on the bit rate.
- After being modified, the checksum must be calculated anew for each SRDO accordingly and then stored in the *Object Dictionary* (see *Index 13FFh*).
- After having modified the configuration a read-back must be carried out for verification.
- After verification of the data, the configuration can be defined as valid (see Object *Dictionary, Index 13FE, Configuration Valid*).

The following items are to be observed at the plausibility test of the configuration:

- The SRVT must be lower than the SCT.
- The COB-ID in the message using the normal data must always be an odd number.
- The COB-ID in the message using the inverted data must always be an even number.

For the significance of the set parameters, the designer of the total system is responsible (see COB-ID Bereich DS304, see [4]).

The plausibility test is carried out when writing the value A5h to *Configuration Valid*. The writing is successful only if the test has been successfully completed. The plausibility test is done for all the SRDOs activated (*Information Direction* unequal with 0).

Important note:
The SRDOs are sent out only if the configuration is valid.

6.5 Emergency Object (EMCY)

EMCY objects are sent when an error occurs. *EMCY* objects contain an *Emergency Error Code,* the contents of an *Error register* as well as a *Manufacturer specific Error Field*. If a notified error is eliminated or disappears, this is also notified by a special *EMCY* object.

An emergency message will be sent, if an error occurs or if this error disappears. The message is structured as follows:

Error	Emergency ErrorCode	Manufacturer SpecificErrorField	Error Category
No Error	0000h	0000h	Error eliminated
Error while loading User Set-up	FF00h	0001h	Manufacturer-specific error (bit 7)
Controller error	FF00h	0002h	Manufacturer-specific error (bit 7)
CAN Error passive	8120h	0003h	Communication error (bit 4)
Recovered from Bus-off	8140h	0004h	Communication error (bit 4)

The *EMCY* object has the pre-set ID 80h+*Node ID* and can be changed in the *Object Dictionary* (see *Object Dictionary, Index 1014h*).

6.6 Heartbeat

Using the *Heartbeat Protocol* the individual nodes can be monitored. CANopen differentiates between the following functions:

1. *Heartbeat Producer*
   sends a *Heartbeat o*bject in cyclical intervals.
2. *Heartbeat Consumer*
   monitors the sending of certain *Heartbeat* objects.

The cycle time can be adjusted in the *Object Dictionary* to milliseconds. If a time interval of 0 is specified, this means "*Heartbeat* not active".

With the *Heartbeat* object the status of the *Heartbeat Producers* is also transmitted in the form of bytes.

Meaning of the Heartbeat object contents

Value	Status	NOTE
0	*BOOTUP*	The device has booted up.
4	*STOPPED*	The device has been stopped.
5	*OPERATIONAL*	The device is working normally.
127	*PRE-OPERATIONAL*	The device does not send any *PDOs*, but can modify *SDOs*.

The HAT CANopen Safety can operate as a *Heartbeat Producer*. The ID of the *Heartbeat* is 700h + *Node-ID*. The time has been pre-set to 0 (not active) and can be modified (see *Object Dictionary, Index 1017h*).

6.7 Network Management Services (NMT)

NMT objects are used to start, stop or reset devices. CANopen differentiates between the following functionalities:

1. *NMT Master*
   controls other nodes.
2. *NMT Slave*
   is controlled by a *Master*.

In a CANopen network only one *NMT* object exists with the Identifier 0. Two bytes are always transmitted. The first byte contains the *Command Specifier*, which represents the command, the second byte contains the *Node ID* of the node which carries out this command. A value of 0 indicates that this command is valid for all nodes. The following commands are possible:

NMT commands

1. ***Start Remote Node***
   The node changes to the *Operational* condition.
2. ***Stop Remote Node***
   The node changes to the *Stopped* condition.
3. ***Enter Pre-Operational***
   The node changes to the *Pre-Operational* condition.
4. ***Reset Node***
   The "Device Profile Specific" OD range is reset, the Baud rate is, if necessary, re-initialised and then changes to the Reset Communication condition.
5. ***Reset Communication***
   The communication unit of the node is reset and then the node changes to the *Pre-Operational* condition.

The HAT CANopen Safety operates as a *NMT Slave* and supports all the *NMT* services.

6.8 Boot Up Protocol

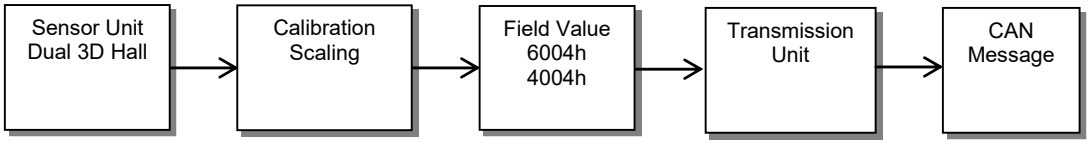
After initialisation, an NMT slave will send a *Boot Up* object. In principle, this is no different to a *Heartbeat* object with the status 0.

In the HAT CANopen Safety this function is implemented.

7 Data flow in the HAT CANopen (Safety)

The following figures show the flow of data within the HAT CANopen Safety, and the relevant indices of the *Object Dictionary*.

Data flow in the HAT CANopen (Safety)



7.1 Sensor Unit

The sensor unit measures the turning angle at the shaft in relation to the housing flange. The type of sensor (*Encoder)* is stored in *Sensor Type*.

7.2 Calibration & Scaling

The sensor values are processed in the *Calibration & Scaling* menu point and made available as a *FieldValue*.

7.3 Transmission Unit

If one of the following events occurs, then depending on the preset *TransmissionType,* the value of the PDO is sent.

1. The *EventTimer* has expired (cyclical transmission).
2. One or more *Sync* objects have been received (synchronous transmission).

8 The Object Dictionary

8.1 Set-up of the Object Dictionary

All the data are stored in the *Object Dictionary*. The entries supported by the HAT CANopen Safety are listed in the following chapters. The index is always shown in hexadecimal notation, according to the specification, without the hexadecimal representation being shown extra. For each entry, the access type must also be specified. CANopen differentiates between the following Access Types:

Access Types for the *Object Dictionary*

1. const
   Read-only, and always delivers the same value.
2. read only
   Read-only, however, the value can be changed during operation.
3. write only
   The entry is write-only.
4. read write
   The entry can be read and written.

CANopen differentiates between the following areas of the data dictionary:

Areas of the Object Dictionary:

1. Index 0000h .. 1FFFh: Communication profile specific entries
   Settings which apply to all CANopen devices. These entries are defined in DS301 and DS302.
2. Index 6000h .. 9FFFh: Device profile specific entries
   Device-specific data which are defined in a Draft Standard.
   The HAT CANopen has implemented the device profile DS406.
3. Index 2000h .. 5FFFh: Manufacturer-specific entries
   Manufacturer-specific, additional data which are not defined in any specification.

8.2 Structure of the device-specific part according to DS406

The HAT CANopen Safety implements DS406. This describes the behaviour and the functionality of angle sensors *(*Encoders [6]*)*.

9 Entries in the Object Dictionary

Listed below are the functionalities implemented by the HAT CANopen Safety. A detailed description of the entries can be found in profiles [1] and [6].

9.1 Communication Profile Specific Entries (DS301)

9.1.1 Index 1000h: DeviceType (read only)

Contains the number of the device profile being used:
406 (0196h) - *Device profile number*
As well as the profile-specific extension:
1 (0001h) - *Single-turn absolute rotary encoder*

9.1.2 Index 1001h: ErrorRegister (read only)

Contains the actual error condition (see *EMCY,* and [1]).

9.1.3 Index 1002h: Manufacturer status register (read only)

Always returns 0

9.1.4 Index 1003h: Pre-defined error field (read only)

Errors are made available, which have occurred in the CANopen device and which were signalised via *emergency object* (see CiA 301)

9.1.5 Index 1005h: SyncMessageIdentifier (read write)

Use this to adjust the *COB ID* for the *SYNC o*bject.

9.1.6 Index 1008h: ManufacturerDeviceName (const)

Provides the device name as a character string. ("HAT 1000" or "HAT 3000").

9.1.7 Index 1009h: ManufacturerHardwareVersion (const)

Provides the hardware version as a character string (e.g. "01.01").

9.1.8 Index 100Ah: ManufacturerSoftwareVersion (const)

Provides the software version as a character string (e.g. "01.07"). The first two characters indicate the version, the last two characters indicate the revision status.

9.1.9 Index 1010h: StoreParameters

By entering the character string "save" (65766173h) the current settings are transferred to the non-volatile memory.
The HAT CANopen Safety does not automatically store settings if modified, modifications are stored only when requested.

WARNING: Any changed settings must be saved explicitly using *StoreParameters*, otherwise they will be lost when the instrument is switched off or when the *NMT* commands *Reset Node* and *Reset Communication* are carried out.

CANopen has the option of restoring different parameter areas with the help of various *Sub indeces*.
The sub indeces 1, 2, 3 and 4 are supported.

For further information, please see profile [1].

Warning: The Baud rate and Node ID settings remain unchanged with *StoreAllParameters*.

Sub indices used:

0: LargestSubindexSupported (read only)
1: StoreAllParameters (read write)
2: StoreCommunicationParameters (read write) (index from 1000h to 1FFFh)
3: StoreApplicationParamteres (read write) (index from 6000h to 9FFFh)
4: StoreLssParameters (read write) (index von 2000h to 20FFh)

9.1.10 Index 1011h: RestoreDefaultParameters

By entering the character string "load" (64616F6Ch) the factory settings are transferred into the non-volatile memory.
However, the HAT CANopen Safety goes on working with the actual settings until switched off or until the commands *Reset Node* and *Reset Communication* are performed.

CANopen has the option of restoring different parameter areas with the help of various *Sub indeces*.
The sub indeces 1, 2, 3 and 4 are supported.

For further information, please see profile [1].

Warning: The Baud rate and Node ID settings remain unchanged with *RestoreAllParameters*.

Sub indices used:

0: LargestSubindexSupported (read only)
1: RestoreAllParameters (read write)
2: RestoreCommunicationParamters (read write) (index from 1000h to 1FFFh)
3: RestoreApplicationParameters (read write) (index from 6000h to 9FFFh)
4: RestoreLssParameters (read write) (index from 2000h to 20FFh)

9.1.11 Index 1014h: CobIdEmergencyMessage (read write)

Use this to adjust the *COB ID* for the *EMCY* object (see *EMCY*).

9.1.12 Index 1017h: ProducerHeartbeatTime (read write)

Use this to adjust the *Heartbeat* time to milliseconds. The value 0 indicates that this function is not active (see *Heartbeat*).

9.1.13 Index 1018h: IdentityObject

The Identity Object identifies the HAT CANopen (Safety). The identification consists of four 32bit numbers. The combination of these 4 numbers produces a device ID which is unique worldwide.

Sub indices used:

0: LargestSubindexSupported (read only)
1: VendorID (read only)
Unique manufacturer code (DAh for HYDAC ELECTRONIC GmbH)
2: ProductCode (read only)
Hydac Electronic product code (e.g.: 925010)
3: RevisionNumber (read only)
Revision number of the device
4: SerialNumber (read only)
Serial number of the device

9.1.14 Index 1029h: Error behaviour

Sub indices used:

0: No. of Error Classes (read only)
1: Communication Error (read write)
0 : Change to NMT state Pre-operational
1 : No change of the NMT state
2 : Change to NMT state Stopped
2: Specific Error Class (read write)
0 : Change to NMT state Pre-operational
1 : No change of the NMT state
2 : Change to NMT state Stopped

9.1.15 Index 1301h: SRDO communication parameter

Contains the communication settings of the first safety relevant data object.

Sub indices used:

0: LargestSubindexSupported (read only)
1: Information direction (read write)
2: Refresh-time or SCT (read write)
3: Validation time (read write)
4: Transmission type (read only)Transmission type (read only)
5: COB-ID 1 (read write)
6: COB-ID 2 (read write)

9.1.16 Index 1381h: SRDO mapping parameter

Contains the mapping settings of the first safety relevant data object.

Sub indices used:

0: LargestSubindexSupported (read write)
1: Mapping Object 1 Data Not Inverted (read write)
2: Mapping Object 1 Data Inverted (read write)
3: Mapping Object 2 Data Not Inverted (read write)
4: Mapping Object 2 Data Inverted (read write)

5: Mapping Object 3 Data Not Inverted (read write)
6: Mapping Object 3 Data Inverted (read write)
7: Mapping Object 4 Data Not Inverted (read write)
8: Mapping Object 4 Data Inverted (read write)
9: Mapping Object 5 Data Not Inverted (read write)
10: Mapping Object 5 Data Inverted (read write)
11: Mapping Object 6 Data Not Inverted (read write)
12: Mapping Object 6 Data Inverted (read write)
13: Mapping Object 7 Data Not Inverted (read write)
14: Mapping Object 7 Data Inverted (read write)
15: Mapping Object 8 Data Not Inverted (read write)
16: Mapping Object 8 Data Inverted (read write)

When supplied, the HAT CANopen Safety has the following entries:

Index 1381h		
Subindex	**Content**	**Meaning**
0	2	Two values are transferred in the *SRDO.*
1	60040020h	The first value in the *SRDO* is the value of *Index 6004h*, S*ubindex* 0 with a width of 20h (=32 bits)
2	40040020h	The second value in the *SRDO* is the value of Index 4004h, *Subindex* 0 with a width of 20h (=32 bits)

9.1.17 Index 13FEh: Configuration Valid (read write)

This entry does not serve for validation of the SRDO configuration. A plausibility test is triggered when writing the value A5h. If the test is successful the value is accepted. Otherwise, the value is set to 0.

9.1.18 Index 13FFh: Safety Configuration Checksum

Contains the checksums of the SRDO configurations.

Sub indices used:

0: LargestSubindexSupported (read only)
1: Checksum-1 (read write)

9.1.19 Index 1800h: TPDO communication parameter

This entry determines the parameters for the PDO transmission. In detail these are:

Parameters for PDO transmission

1. *COB ID*
   Determines the identifier for the PDO. The highest value bit (Bit31) of the entry no longer belongs to the ID and has the meaning "*disable PDO*". If this bit is set, then transmission of the PDO is disabled.
2. *Transmission Type*
   Determines the transmission type.
   Values between 0 and 240 mean synchronous transmission. The figure represents the number of SYNC objects which have to be received before the PDO is sent.
   The value 254 indicates a manufacturer-specific transmission and the value 255 a device-profile-specific transmission. With 254 and 255 the PDO is sent cyclically, providing a time (*Event Time*) other than 0 is set.

3. *Inhibit Time*
   defines the minimum interval for the PDO if FEh and FFh were chosen as transmission type. The value is defined as a multiple of 100µs. The value 0 switches off the *inhibit time*.
4. *Reserved – of no significance*
5. *Event Time*
   Determines the cycle time for asynchronous transmissions for Transmission Types 254 and 255 in milliseconds. The value 0 denotes no time-controlled transmission.

Sub indices used:

0: LargestSubindexSupported (read only)
1: COBIDUsedByPDO (read write)
2: TransmissionType (read write)
3: InhibitTime (read write)
4: Reserved
5: EventTimer(read write)

9.1.20 Index 1A00h: TPDO mapping parameter

Use this entry to determine which data is transferred with the PDO. Subindex 0 indicates the amount of data in the PDO. The index and subindex as well as the number of bits of the first date are stored in subindex 1, this also applies for the sub indeces 2 to 8.

When supplied, the HAT CANopen Safety has the following entries:

Index 1A00h **-- Deactivated by TPDO communication parameter --**		
Subindex	**Content**	**Meaning**
0	1	One value is transferred with the *PDO*.
1	60040020h	The first value in the *PDO* is the value of *Index 6004h*, *Subindex* 0 with a width of 20h (=32 bits)

Sub indices used:

0: NumberOfMappedApplicationObjectsInPDO (read write)
Values 0 to 8 are permitted.
This means: no PDO is transmitted, or one PDO with up to 8 values is transmitted.

ObjectToBeMapped (1 to 8) (read write)

In HAT CANopen Safety the following values are permitted:

10010008h = Error Register 8bit width
10020020h = Manufacturer Status Register 32bit width
40040020h = Process value Position Value with 32 bit width **inverted**(DS406 [6])
60040020h = Process value Position Value with 32 bit width (DS406 [6])
65030010h = Alarms with 16 bit width (DS406 [6])

9.1.21 Index 1F80h: NMT-Startup (read / write)

If bit 2 is set, status is changed automatically to "Operational" state immediately after reaching "Pre-Operational" state (DS302).
Permitted values are: **08h** and **0Ch**

9.2 Device Profile Specific Entries (DS406)

9.2.1 Index 6000h: Operating Parameter (read only)

This entry contains the parameters for the output of the *Position Value* ([6]).
Code sequence
Commissioning diagnostic control
Commissioning bit
Scaling function control
Measuring direction

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
msp	msp	msp	msp	r	r	r	r	r	r	r	r	md	sfc	cdc	cs
MSB															LSB

Signal	Value	Definition
cs	0b 1b	Code sequence = CW Code sequence = CCW
cdc	0b 1b	Commissioning diagnostic control not supported Commissioning diagnostic control supported
sfc	0b 1b	Scaling function control disabled Scaling function control enabled
md	0b 1b	Measuring direction forward Measuring direction backwards
r		Reserved for further use
msp	0b 1b	Manufacturer-specific function disabled Manufacturer-specific function enabled

9.2.2 Index 6001h: Measuring units per revolution (read only)

This entry indicates how many measured units per rotation are counted ([6]).

9.2.3 Index 6002h: Total measuring range in measuring units (read only)

This entry indicates how big the measuring range is in measured units ([6]).

9.2.4 Index 6003h: Preset value (read write)

This entry contains the offset of the measuring system in measured units (*Preset value*, [6])

9.2.5 Index 6004h: Position value (read only)

This entry contains the measured position value (*Position value*, [6])

9.2.6 Index 6500h: Operating Status (read only)

Corresponds with Index 6000h [6].

9.2.7 Index 6501h: Single-turn resolution and Measuring step (read only)

Indicates how many measuring steps correspond with one entire rotation [6].

9.2.8 Index 6503h: Alarms (read only)

Returns 1 in case of a detected measuring error, otherwise 0 [6].

9.2.9 Index 6504h: Supported alarms (read only)

Always returns 1 [6].

9.2.10 Index 6505h: Warnings (read only)

Always returns 0 [6].

9.2.11 Index 6506h: Supported warnings (read only)

Always returns 0 [6].

9.2.12 Index 6507h: Profile and software version (read only)

Returns the device software and the profile version [6].

9.2.13 Index 6508h: Operating time (read only)

Operation hours counter, not implemented in the HAT, always returns FFFFFFFFh.

9.2.14 Index 6509h: Offset value (read only)

This index contains the offset value for the position [6].

9.2.15 Index 650Ah: Module identification (read only)

This index contains the manufacturers specifications on the offset and the minimum and maximum position [6].

Sub indices used:

0: NumberOfEntries (read only)
1: Manufacturer offset value (read only)
2: Manufacturer min position value (read only)
3: Manufacturer max position value (read only)

9.3 Manufacturer Specific Entries

9.3.1 Index 2001h: Node-ID

The *Node ID* is stored in this index.

Sub indices used:

4: NumberOfEntries (read only)
5: CurrentNodeId (read only)
6: DemandedNodeId (read write)

In order for the new *Node ID* to be effective, the command *StoreLssParameters* must first be transmitted and the node must be restarted afterwards.

9.3.2 Index 2002h: Baudrate

The Baud rate is stored in this index.
The allocation of the entry to the Baud rate complies with DS305, *Layer Setting Services and Protocols*.

Entry	0	1	2	3	4	5	6	7	8
Baud rate	1000 kbit/s	800 kbit/s	500 kbit/s	250 kbit/s	125 kbit/s	**reserved**	50 kbit/s	20 kbit/s	10 kbit/s

In order for the new *Baud rate* to be effective, the command *StoreLssParameters* must first be transmitted and the node must be restarted afterwards.

Sub indices used:

0: NumberOfEntries (read only)
1: CurrentBaudrate (read only)
2: DemandedBaudrate (read write)

9.3.3 Index 4006h: Position Value (read only) Position inverted

Identical with Index 6004h, but **inverted bitwise**.

9.3.4 Further indeces within the range of 2000h to 5FFFh (reserved)

The indices within this range contain **important factory settings which must not be modified by the customer!**
Warning: Each **Intervention** may lead to **malfunction** of the angle sensor!

10 Layer setting services (LSS) and protocols

The LSS services and protocols, documented in CiA DS305 V2.2 (see [4]), are used for the inquiry or the modification of certain parameters of the Data Link Layer and the Application Layer of a LSS slave by a LSS master via the CAN network.

The following parameters are supported:

- Node-ID
- Baud rate
- LSS address, according to identity object 1018h

Access to the LSS slave thereby is made by its LSS address, consisting of:

- Vendor ID
- Product Code
- Revision number
- Serial number

The measuring system supports the following services:

Switch mode services

- Switch mode selective
  - To address a specific LSS slave
- Switch mode global
  - To address the total of LSS slaves

Configuration services

- Configure Node ID
  - Configure Node ID
- Configure bit timing parameters
  - Configure Baud rate
- Activate bit timing parameters
  - Activate Baud rate
- Store configured parameters
  - Store configured parameters

Inquiry services

- Inquire LSS address
  - Inquire LSS address
- Inquire Node ID
  - Inquire Node ID

Identification services

- LSS identify remote slave
  - Identification of LSS slaves within a certain array
- LSS identify slave
  - Response of all LSS slaves to the previous command
- LSS identify non-configured remote slave
  - Identification of non-configured LSS slaves, Node ID = FFh
- LSS identify non-configured slave
  - Response of all LSS slaves to the previous command

10.1 Finite state automaton, FSA

The FSA is equivalent to a state machine and defines the behaviour of a LSS slave. This state machine is controlled by LSS COBs produced by the LSS master or NMT COBs generated by a NMT master or local NMT state transitions.

The LSS Modes support the following states:
(0) Initial: Pseudo-State, shows the activation of the FSAs
(1) LSS waiting: Support for all services as indicated below
(2) LSS configuration: Support for all services as indicated below
(3) Final: Pseudo-State, shows the deactivation of the FSAs

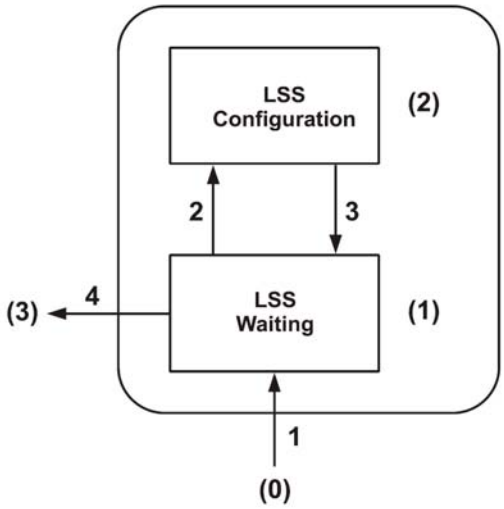


Figure 1: LSS Modes

State behavior of the supported services

Services	Waiting	Configuration
Switch mode global	Yes	Yes
Switch mode selective	Yes	No
Activate bit timing parameters	No	Yes
Configure bit timing parameters	No	Yes
Configure Node ID	No	Yes
Store configuration	No	Yes
Inquire LSS address	No	Yes
LSS identify remote slave	Yes	Yes
LSS identify slave	Yes	Yes
LSS identify non-configured remote slave	Yes	Yes
LSS identify non-configured slave	Yes	Yes

LSS FSA state transitions

Transition	Events	Actions
1	Automatic transition after initialisation when entering either the state NMT PRE OPERATIONAL state or NMT STOPPED state, or NMT RESET COMMUNICATION state with Node ID = FFh.	None
2	LSS 'switch state global' command with parameters 'configuration_switch' or 'switch state selective' command	None
3	LSS 'switch state global' command with parameter 'waiting_switch'	None
4	Automatic transition when an invalid Node ID has been changed and the new Node ID could be successfully stored in non-volatile memory AND the state of LSS waiting has been requested.	None

Once the LSS FSA further state transitions of the NMT FSA on NMT PRE into OPERATIONAL STOPPED and vice versa, this will not lead to re-entry into the LSS FSA.

10.2 Transmission of LSS services

By means of LSS services, the LSS master requests the particular services to be performed by the LSS slave. Communication between LSS master and LSS slave is performed by means of implemented LSS protocols.
Similar as in the case of SDO transmission, two COB IDs for sending and receiving are used in this case as well:

COB ID	Meaning
7E4h	LSS-Slave → LSS-Master
7E5h	LSS-Master → LSS-Slave

Table 1: COB IDs for LSS services

10.2.1 LSS message format

The data field of a CAN message being max. 8 bytes long is used by a LSS service as follows:

CS	Data						
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Table 2: LSS message

Byte 0 contains the ***Command-Specifier*** (CS), afterwards 7 bytes of data are following.

10.3 Switch mode protocols

10.3.1 Switch mode global protocol

The given protocol has implemented the *Switch mode global service* and controls the state behavior of the LSS slave. By means of the LSS master all LSS slaves in the network can be switched to *Waiting Mode* or *Configuration Mode*.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Mode**	Reserved by CiA					
7E5h	04	0 = Waiting Mode 1 = Configuration Mode						

10.3.2 Switch mode selective protocol

The given protocol has implemented the *Switch mode selective service* and controls the state behavior of the LSS slave. By means of the LSS master only this LSS slave in the network can be switched to *Configuration Mode*, whose LSS address attributes correspond with the LSS address.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Vendor ID**				Reserved by CiA		
7E5h	64	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Product-Code**				Reserved by CiA		
7E5h	65	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Revision number**				Reserved by CiA		
7E5h	66	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Serial number**				Reserved by CiA		
7E5h	67	LSB			MSB			

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E4h	68							

10.1 Configuration protocols

10.1.1 Configure Node ID Protocol

The given protocol has implemented the *Configure NMT-address service*. By means of the LSS master the Node ID of a single LSS slave in the network can be configured. Only one LSS slave may be switched to *Configuration Mode* for this purpose. For storage of the new Node ID the *Store configuration protocol* must be transmitted to the LSS slave. To activate the new Node ID the NMT service *Reset Communication (82h)* must be addressed.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Node ID**	Reserved by CiA					
7E5h	17	1…127 and 255						

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Error Code**	**Spec. Error**	Reserved by CiA				
7E4h	17							

Error Code
0: Protocol successfully completed
1…254: Reserved
255: application specific error occurred

Specific Error
if Error Code = 255 --> application specific error occurred,
otherwise reserved by CiA

10.1.2 Configure bit timing parameters protocol

The given protocol has implemented the *Configure bit timing parameters service*. By means of the LSS master the Baud rate of a single LSS slave or of all LSS slaves in the network can be configured. For the storage of the new Node ID the *Store configuration protocol* must be transmitted to the LSS slave.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Table Selector**	**Table Index**	Reserved by CiA				
7E5h	19	0	0..8					

LSS-Slave --> LSS-Master

0	1	2	3	4	5	6	7
COB ID	**CS**	**Error Code**	**Spec. Error**	Reserved by CiA			
7E4h	19						

Table Selector
0: Standard CiA Baud rate table

Table Index
0: 1 Mbit/s
1: 800 kbit/s
2: 500 kbit/s
3: 250 kbit/s
4: 125 kbit/s
5: **reserved**!
6: 50 kbit/s
7: 20 kbit/s
8: 10 kbit/s

Error Code
0: Protocol successfully completed
1: selected Baud rate not supported
2…254: Reserved
255: application specific error occurred

Specific Error
if Error Code = 255 --> application specific error occurred,
otherwise reserved by CiA

10.1.3 Activate bit timing parameters protocol

The given protocol has implemented the *`Activate bit timing parameters service`* and activates the Baud rate defined via *`Configure bit timing parameters protocol`* in all LSS Slaves in the network being in *`Configuration Mode`*.

LSS-Master --> LSS-Slave

0	1	2	3	4	5	6	7
COB ID	**CS**	**Switch Delay [ms]**		Reserved by CiA			
7E5h	21	LSB	MSB				

Switch Delay
The parameter *`Switch Delay`* defines the length of two delay periods (D1, D2) with equal length. These are necessary to avoid operating the bus with differing Baud rate parameters. After the expiration of the time D1 and after an individual processing duration, the switching is performed internally inside the LSS slave. After the expiration of the time D2, the LSS slave responds with CAN messages and the newly configured Baud rate.

The following applies:
Switch Delay > longest occurring processing duration of a LSS slave

10.1.4 Store configuration protocol

The given protocol has implemented the *Store configured parameters service*. By means of the LSS master the configured parameters of a single LSS slave in the network can be stored into the non-volatile memory. Only one LSS slave may be switched to *Configuration Mode* for this purpose.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E5h	23							

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Error Code**	**Spec. Error**	Reserved by CiA				
7E4h	23							

Error Code
0: Protocol successfully completed
1: *Store configuration* not supported
2…254: Reserved
255: application specific error occurred

Specific Error
if Error Code = 255 --> application specific error occurred,
otherwise reserved by CiA

10.2 Inquire LSS address protocols

10.2.1 Inquire Identity Vendor ID protocol

The given protocol has implemented the *Inquire LSS address service*. By means of the LSS master the Vendor ID of a single LSS slave in the network can be read out. Only one LSS slave may be switched to *Configuration Mode* for this purpose.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E5h	90							

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Vendor-ID** (= Index 1018h:01)				Reserved by CiA		
7E4h	90	LSB			MSB			

10.2.2 Inquire Identity Product Code Protocol

The given protocol has implemented the *`Inquire LSS address service`*. By means of the LSS master the manufacturer's product name of a single LSS slave in the network can be read out. Only one LSS slave may be switched to *`Configuration Mode`* for this purpose.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E5h	91							

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Product Code** (= Index 1018h:02)				Reserved by CiA		
7E4h	91	LSB			MSB			

10.2.3 Inquire Identity Revision-Number protocol

The given protocol has implemented the *`Inquire LSS address service`*. By means of the LSS master the Identity Revision Number of a single LSS slave in the network can be read out. Only one LSS slave may be switched to *`Configuration Mode`* for this purpose.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E5h	92							

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Revision Number** (= Index 1018h:03)				Reserved by CiA		
7E4h	92	LSB			MSB			

10.2.4 Inquire Identity Serial Number protocol

The given protocol has implemented the *Inquire LSS address service*. Via the LSS master the Serial Number of a single LSS slave in the network can be read out. Only one LSS slave may be switched to *Configuration Mode* for this purpose.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E5h	93							

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Serial Number** (= Index 1018h:04)				Reserved by CiA		
7E5h	93	LSB			MSB			

10.2.5 Inquire Node ID protocol

The given protocol has implemented the *Inquire Node ID service.*
Via the LSS master the Node ID of a single LSS slave in the network can be read out. Only one LSS slave may be switched to *Configuration Mode* for this purpose.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	Reserved by CiA						
7E5h	94							

LSS-Slave --> LSS-Master

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Node ID**	Reserved by CiA					
7E4h	94	1…127 and 255						

Node-ID
Corresponds to the Node ID of the selected device. If the Node ID has been recently changed by *the Configure Node ID service*, the original Node ID is returned. Only after implementation of the NMT Service *Reset Communication* (82h), the current Node ID is returned.

10.3 Identification Protocols

10.3.1 LSS identify remote slave protocol

The given protocol has implemented the *LSS identify remote slaves service*. By means of the LSS master LSS slaves in the network can be identified within a certain range. All LSS slaves with matching Vendor ID, Product Code, Revision No. - area and Serial No. - area, respond by the *LSS identify slave protocol*.

LSS-Master --> LSS-Slave

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Vendor ID**				Reserved by CiA		
7E5h	70	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Product-Code**				Reserved by CiA		
7E5h	71	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Revision-Number-Low**				Reserved by CiA		
7E5h	72	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Revision-Number-High**				Reserved by CiA		
7E5h	73	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Serial-Number-Low**				Reserved by CiA		
7E5h	74	LSB			MSB			

	0	1	2	3	4	5	6	7
COB ID	**CS**	**Serial Number High**				Reserved by CiA		
7E5h	75	LSB			MSB			

10.3.2 LSS identify slave protocol

The given protocol has implemented the *LSS identify slave service*. All LSS slaves with matching LSS address attributes given in the *LSS identify remote slaves protocol*, respond by this protocol.

LSS-Slave --> LSS-Master

0	1	2	3	4	5	6	7

COB ID	**CS**	Reserved by CiA
7E4h	79	

10.3.3 LSS identify non-configured remote slave protocol

The specified protocol has implemented the *LSS identify non-configured remote slave service*. By means of the LSS Master all non-configured LSS Slaves (Node ID = FFh) in the network are identified. The related LSS Slaves respond by the *LSS identify non-configured remote slave protocol*.

LSS-Slave --> LSS-Master

0	1	2	3	4	5	6	7

COB ID	**CS**	Reserved by CiA
7E4h	76	

10.3.4 LSS identify non-configured slave Protocol

The specified protocol has implemented the LSS *identify non-configured slave service*. All LSS slaves having an invalid Node ID (FFh) respond by this protocol after having completed the *LSS identify non-configured slave protocol.*

LSS-Slave --> LSS-Master

0	1	2	3	4	5	6	7

COB ID	**CS**	Reserved by CiA
7E4h	80	

11 Connection

The connection can be carried out by means of the enclosed device specific pin assignment (see user manual HAT CANopen Safety (the user manual is included in the delivery of HAT CANopen Safety).

11.1 Switching on the supply voltage

After connection and all settings have been carried out, the supply voltage can be switched on.

After POWER ON and finishing the initialisation, the measuring system enters the PRE-OPERATIONAL state. This status is acknowledged by the Boot-Up message "**COB ID 700h+Node ID**". If the measuring system detects an internal error, an emergency message containing the error code is transmitted (see 6.5).

In PRE-OPERATIONAL state, at first, only a parameter setting via Service Data Objects is possible. But it is possible to configure PDOs with the help of SDOs. If the measuring system has entered the OPERATIONAL state, a transmission of PDOs is possible as well (see 6.7).

11.2 Setting the Node ID and Baud rate by means of LSS services

11.2.1 Configuration of the Node ID, sequence

Assumption:

- LSS address unknown
- only one LSS slave should be in the network
- the Node ID 12 dec. shall be set

Course of action:

- Switch LSS-Slave to *Configuration Mode* using command Specifier 04 *Switch mode global protocol*, Mode = 1.
  (see 0)
- Perform command Specifier 17 *Configure NMT-Address protocol*, Node-ID = 12. (see 10.1.1)
  --> Wait for feedback and check successful execution,
  --> Error Code = 0.

- Carry out command Specifier 23 *Store configuration protocol* .
  --> Wait for feedback and check successful execution,
  --> Error Code = 0
  (see10.1.4).
- Switch off the supply voltage of the LSS slave, then switch on again. Now the new configuration is activated.

11.2.2 Configuration of the Baud rate, sequence

Assumption:

- LSS address unknown
- only one LSS slave should be in the network
- the Baud rate 125 kbit/s shall be set

Course of action:

- Perform NMT command Specifier *Stop Remote Node* (02h) in order to switch LSS-Slave into *Stopped state* . The LSS slave shouldn't send any more CAN-messages--> Heartbeat switched off
  (see 6.7).
- Switch LSS-Slave to *Configuration Mode* using command Specifier *Switch mode global protocol*, Mode = 1.
  (see 10.3.1).
- Perform command Specifier 19 *Configure bit timing parameters protocol*, Table Selector = 0, Table Index = 4
  --> wait for feedback and check successful execution,
  --> Error Code = 0
  (see 10.1.2).
- Carry out command Specifier 21 *Activate bit timing parameters protocol* in order to activate the new Baud rate.
  (see 10.1.3).
- Carry out Command Specifier 23 *Store configuration protocol*.
  --> Wait for feedback and check successful execution,
  --> Error Code = 0
  (see10.1.4).
- Switch off the supply voltage of the LSS slave, then switch on again. Now the new configuration is activated.

12 Commissioning

12.1 CAN interface

The CAN bus interface is defined by the international standard ISO/DIS 11898 and specifies the two lowest layers of the ISO/DIS CAN Reference Model.

The conversion of the measuring system information to the CAN message format (CAN 2.0A) is done by a CAN-controller. The function of the CAN-controller is controlled by a watchdog.

The CANopen Communication Profile (CiA standard DS301, see [1]) is a subset of CAN Application Layer (CAL) and describes how the services are used by the devices. The CANopen Profile allows the definition of device profiles for decentralized I/O.

The measuring system with CANopen protocol supports the Device Profile for angle sensors (CiA Draft Standard 406, Version 1.3.0, see [6]).

The communication functionality and objects, used in the measuring device profile, are described in an EDS-File (Electronic Data Sheet). When using a CANopen Configuration Tool (e.g.: CANSETTER), the user can read out the objects of the measuring system (SDOs) and program the functionality.

12.2 EDS file

The EDS (electronic datasheet) contains all information on the measuring system-specific parameters and the measuring system’s operating modes. The EDS file is integrated using the CANopen network configuration tool to correctly configure or operate the measuring system.

The EDS file matching the device can be identified by the device name and by the device related major revision number in its file name.

The file including the its CANopen Safety interface description can be found and downloaded from our homepage.

http://www.hydac.com/uk-en/products/sensors/show/Material/index.html

HYDAC ELECTRONIC GMBH
Hauptstr. 27
D-66128 Saarbruecken
Germany

Web: www.hydac.com
E-Mail: electronic@hydac.com
Tel.: +49 (0)6897 509-01
Fax.: +49 (0)6897 509-1726

HYDAC Service
For enquiries regarding repairs, please contact HYDAC Service.

HYDAC SERVICE GMBH
Hauptstr. 27
D-66128 Saarbruecken
Germany

Tel.: +49 (0)6897 509-1936
Fax: +49 (0)6897 509-1933

NOTE

The information in this manual relates to the operating conditions and applications described. For applications and operating conditions not described. please contact the relevant technical department.

If you have any questions, suggestions, or encounter any problems of a technical nature, please contact your HYDAC representative.