User Manual of the Electronic Data Sheet for Pressure Transmitters with CANopen Interface HDA 4000 CANopen

Firmware
Version 1 Release 2
(Supplement to the Original Manual)

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.

Always keep the manual with the device for immediate reference.
Note that the data on the software technology provided in this manual refers to that available at the time of publication.

If you find any errors while reading the documentation or have additional comments or suggestions, please contact us at:

HYDAC ELECTRONIC GMBH
Technische Dokumentation
Hauptstraße 27
66128 Saarbrücken
-Germany-
Tel.: +49 (0) 6897 / 509 – 01
Fax: +49(0)6897 / 509-1726
E-mail: electronic@hydac.com

We look forward to receiving your input.

“Putting experience into practice”

1 Introduction

The pressure transmitter HDA 4000 CAN complies with the CANopen Standard according to the following profiles:

- DS301, Version 4.02
  Application Layer and Communication Profile
- DS404, Version 1.2
  Device Profile Measuring Devices and Closed-Loop Controllers

This manual describes the functions supported by the HDA 4000 CAN. A basic knowledge of CAN and CANopen is assumed. The exact function is described in the *Draft Standards* DS301 and DS404. 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 HDA 4000 CAN

- Measuring the actual pressure value using:
  - 1kHz sample rate
  - 0.2% accuracy
  - 0.1% resolution
- Conversion of the pressure value into a user-scaleable linear process value.
- Sending the actual process value as a PDO for the following events:
  - Synchronously in relation to received SYNC objects
  - Asynchronously, cyclically, in the range 1 millisecond to 1 minute
  - When the measured value changes in relation to an adjustable differential
  - When an adjustable limit is exceeded
  - When the value falls below an adjustable limit

3 Transmission rates

The HDA 4000 CAN supports the following transmission rates (Baud rates):

- 1 Mbit/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 250 kbit/s and can CAN-Bus (see *Object Dictionary Index 2001*)

4 CAN Frames

The HDA 4000 CAN supports the 11-bit base frames with 11-bit identifier required in the specification. Extended frames with 29-bit identifier are not supported but are tolerated. This means that extended frames are not recognised but neither do they cause errors.

5 Node ID

To operate the HDA 4000 CAN 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, like the transmission rate, and can also be adjusted via the CAN-Bus (see *Object Dictionary Index 2002*) . 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) is filed in an *Object Dictionary* under a specified *Index*. Some 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 HDA 4000 CAN. The HDA 4000 CAN takes on the role of a *Server*, and the device which wants to read or write the data, takes on the role of a *Client*.

To transmit data the HDA 4000 CAN must have a *Receive-SDO* with which it receives data and a *Transmit-SDO* with which it sends data. Sequence of the transmission of data:

<u>Reading from Object Dictionary:</u>

1. An instrument (*Client*) sends the *Receive-SDO* of the HDA 4000 CAN (*Server*).
   In this *SDO* there is an identification to say that the *Object Dictionary* is to be read, as well as the *Index* and *Subindex*.
2. The HDA 4000 CAN (*Server*) sends its *Transmit-SDO*. Here also are the *Index* and the *Subindex*, and the read data.

<u>Writing to the Object Dictionary:</u>

1. A device (Client) sends the Receive-SDO of the HDA 4000 CAN (Server).
   In this SDO there is 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 HDA 4000 CAN (Server) sends its Transmit-SDO. In this there are also 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 Object Directory of the HDA 4000 CAN is configured such that each entry has a maximum of 4 bytes of data, so that all data can be transmitted as *expedited*. In other words, all data can be packed in a single *SDO*, resulting in particularly effective transmissions.

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

<u>*COB-IDs for Service Data Objects*</u>

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 defines what are known as *Process Data Objects*. These contain only the necessary useful data. There are two types of *PDO*:

1. *Transmit-PDOs*
   With these, the measuring instrument can send its measured values.
2. *Receive-PDOs*
   With these, the actuating variables can 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 1A00*).

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 1800*).

Events which result in a PDO being sent:

1. Receipt of a *SYNC* object (synchronous transmission).
2. Expiry of an adjustable cycle time in the range 1 milliseconds to 1 minute (cyclical transmission).
3. The measured value has changed by an adjustable amount compared to the last PDO.
4. The measured value has exceeded an adjustable limit.
5. The measured value has fallen below an adjustable limit.

The HDA 4000 CAN implements a *Transmit-PDO*, which transmits the actual pressure value and the status of the signal input.

The DS404 stipulates the transmission of the actual measured value as a 32-bit value, and the status as an 8-bit value, as the default setting. However, it is possible to change this setting via *PDO-Mapping* so that the measured value is transmitted as a 16-bit value. The transmission of the status can also be disabled by changing the *PDO-Mapping*.

6.3 Synchronisation Object (SYNC)

SYNC objects are used to implement a synchronous data transmission. A *SYNC* object is in principle 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 HDA 4000 CAN provides the functionality of a *SYNC Consumer*. When 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 1005*). In *PDO Transmission Type* the number of received *SYNC* objects which result in a *PDO* being sent, can be set.

6.4 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.

The HDA 4000 CAN sends the following *EMCY* objects:

Emergency Error Code	*Manufacturer specific Error Field*	Technical description
ff00h = *Device specific*	1	Error on signal input
8100h = *Communication Error*	X	Error on the CAN Bus
0000h = *Error Reset or No Error*	0	No error now present

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

6.5 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* in milliseconds. If a time interval of 0 is specified, this indicates "*Heartbeat* not active".

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

<u>Meaning of the Heartbeat object contents</u>

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

The HDA 4000 CAN 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 changed (see *Object Dictionary, Index 1017*).

6.6 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 should carry 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 node is reset and then changes to the *Pre-Operational* condition.
5. *Reset Communicatiuon*
   The communication unit of the node is reset and then the node changes to the *Pre-Operational* condition.

The HDA 4000 CAN operates as a *NMT Slave* and supports all the *NMT* services.

Note: The HDA 4000 CAN does not have an strict division between the communication unit and the application unit. In this respect the two commands *Reset Node* and *Reset Communication* have the same effect.

6.7 Boot Up Protocol

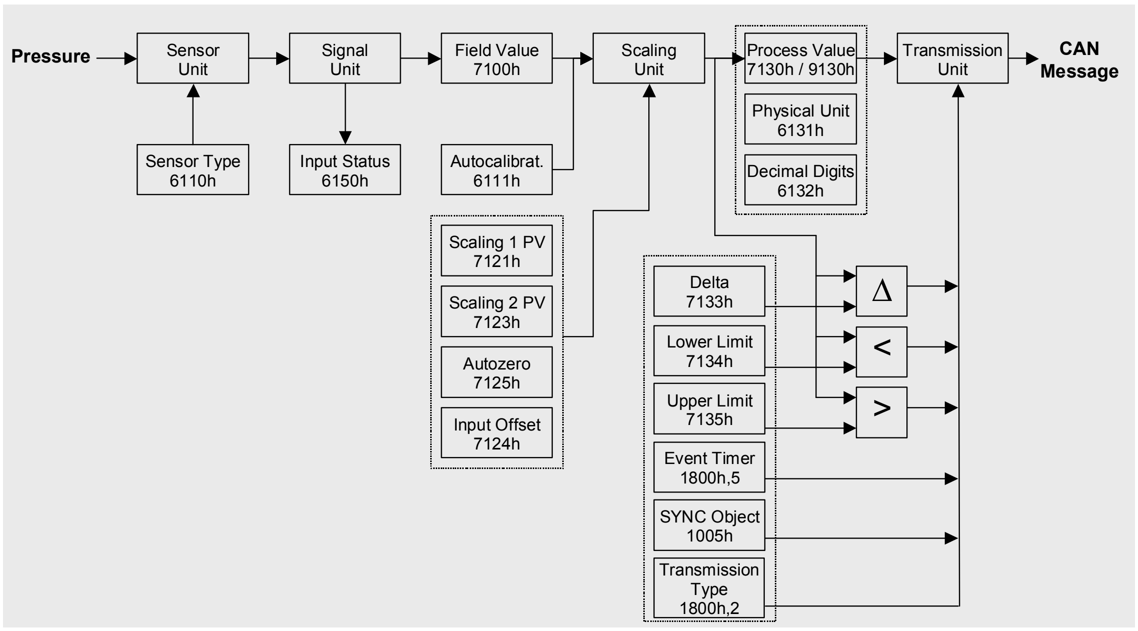
If a *NMT Slave* changes to the *Pre-operational* condition after initialisation, it sends a *Boot Up* object in each case. In principle this is no different to a *Heartbeat* object with the status 0.

On the HDA 4000 CAN this function is implemented.

7 Data flow in the HDA 4000 CAN

The following diagram shows the flow of data within the HDA 4000 CAN, and the relevant indices of the *Object Dictionary*.

Data flow in the HDA 4000 CAN



7.1 Sensor Unit

The sensor unit measures the pressure and converts it into an electrical signal. The type of sensor (*Pressure Transducer)* is stored in *Sensor Type*.

7.2 Signal Unit

The electrical signal is processed in the signal unit, converted to a digital signal and is available as a *FieldValue*. The status of the signal process is stored in *InputStatus*.

7.3 Scaling Unit

The *FieldValue* is then converted in the *Scaling Unit* into a *ProcessValue*.

The indications *PhysicalUnit* and *DecimalDigits* describe how the *ProcessValue* is to be interpreted. If in *PhysicalUnit* for example, the unit is bar and in *DecimalDigits* the value is 1, then a *ProcessValue* of 127 corresponds to the measured value of 12.7 bar.

The conversion of the *FieldValue* into the *ProcessValue* is carried out using the two scaling values *Scaling1PV* and *Scaling2PV*. The first value indicates which process value corresponds to the lower range limit (e.g. 0 bar). The second value indicates which process value corresponds to the upper range limit (e.g. 400 bar).

With *InputOffset* an additional offset can be calculated. With *AutoZero, InputOffset* is set so that the actual process value has the value 0.

With *Autocalibration* an offset error in the signal conditioning can be cleared. The HDA 4000 CAN must be depressurized for this.

7.4 Transmission Unit

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

1. The actual *ProcessValue* has changed by more than *Delta* compared to the last transmitted *Process Value*.
2. The value has fallen below the *LowerLimit*.
3. The Upper*Limit* has been exceeded.
4. The *EventTimer* has expired (cyclical transmission).
5. One or more *Sync* objects have been received (synchronous transmission).

8 The Object Dictionary

8.1 Set-up of the Object Dictionary

As already mentioned, all the data is stored in the *Object Dictionary*. The entries supported by the HDA 4000 CAN are listed in the following chapters. The index is always shown in hexadecimal notation, according to the specification, without the hexadecimal representation being shown specially. 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 from and written to.

CANopen differentiates between the following areas of the data dictionary:

Areas of the Object Dictionary:

1. Index 0 .. 1FFF: *Communication profile specific entries*
   Settings which apply to all CANopen devices. These entries are defined in DS301.
2. Index 6000 .. 9FFF: *Device profile specific entries*
   Device-specific data which are defined in a *Draft Standard*. The HDA 4000 CAN has implemented the device profile DS404.
3. Index 2000 .. 5FFF: *Manufacturer specific entries*
   Manufacturer-specific, additional data which is not defined in any specification.

8.2 Structure of the device-specific part according to DS404

The HDA 4000 CAN implements DS404. This describes the performance and the functionality of measuring devices and closed-loop controllers.

First, some notes on the layout of the *Object Dictionary*:

Multi-channel capability
DS404 is aimed at multi-channel instruments. This means that the number of channels is given for all entries in *Subindex 0* and then the value in the appropriate *Subindex*. The HDA 4000 CAN puts only one channel into effect. Therefore the *Subindex 0* contains the value 1 throughout and the actual measured values or settings are stored in *Subindex 1*.

Data types
DS404 is set up in such a way that each data type has its own section in the *Object Dictionary*:

Index	Data type
6000 to 6FFFh	Floating point or specially encoded data (status, etc.)
7000 to 7FFFh	16 bit integer
8000 to 8FFFh	32 bit integer
9000 to 9FFFh	24 bit integer

Function blocks
DS404 subdivides a device into different function blocks: analogue input block, analogue output block, digital input block, digital output block, controller block and alarm block.
The HDA 4000 CAN has implemented the analogue input part *(Analogue Input Function Block*). These entries are in the area X100h to X1FFh.

Order of entries
As a result of this segmenting, the order of the entries relates only to the value of the last 3 digits of the index. The first digit specifies the data type and is ignored as far as classification is concerned.

9 Entries in the Object Dictionary

Listed below are the functionalities implemented by the HDA 4000 CAN. A detailed description of the entries can be found in profiles [1] and [2].

9.1 Communication Profile Specific Entries (DS301)

9.1.1 Index 1000: DeviceType (read only)

Contains the number of the device profile being used, in this case the number 404, as well as the profile-specific extension, in this case a 2, for support of the analogue input device.

9.1.2 Index 1001: ErrorRegister (read only)

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

9.1.3 Index 1002: ManufacturerStatusRegister (read only)

Contains different error flags.
In the 32bit value the bits have the following meaning:

Bit0: 1= error when reading the Setup Data
Bit1: 1= error when writing to the Setup Data
Bit2: 1= error when reading the factory settings
Bit3: 1= internal error during the A/D conversion
Bit4: 1= internal error in the A/D coupling
Bit5: 1= internal bus error
Bit6: 1= error during the internal program monitoring

Bits 0 to 6 are deleted when the error is eliminated. To find out if an error had occurred, these 7 bits 0x00 to 0x06 are mirrored in bits 0x10 to 0x16. The mirrored bits are not deleted if the error occurred once.

9.1.4 Index 1005: SyncMessageIdentifier (read write)

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

9.1.5 Index 1008: ManufacturerDeviceName (const)

Delivers the device name as a character string. ("HDA4").

9.1.6 Index 1009: ManufacturerHardwareVersion (const)

Delivers the hardware version as a character string (e.g. "0100"). The first two characters indicate the actual device, the last two indicate the status of hardware changes.

9.1.7 Index 100A: ManufacturerSoftwareVersion (const)

Delivers the software version as a character string (e.g. "0101"). The first two characters indicate the version, the last two the revision status.

9.1.8 Index 1010: StoreParameters

By entering the character string "save" the current settings are transferred to the non-volatile memory.
The HDA 4000 CAN does not automatically store settings if they are changed, but 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 saving different parameter areas with the help of various *Subindexes*. The HDA 4000 CAN has not subdivided the parameters and therefore only supports *Subindex 1 (StoreAllParameters)*.

For further information, please see profile [1].

Subindices used:

0: LargestSubindexSupported (read only)
1: StoreAllParameters (read write)

9.1.9 Index 1011: RestoreDefaultParameters

By entering the character string "load" the factory settings are transferred into the non-volatile memory.
However the HDA 4000 CAN goes on working with the actual settings until it is switched off or until the commands *Reset Node* and *Reset Communication* are carried out.

CANopen has the option of restoring different parameter areas with the help of various *Subindexes*.
The HDA 4000 CAN has not subdivided the parameters and therefore only supports *Subindex 1 (RestoreAllDefaultParameters)*.

For further information, please see profile [1].

Warning: The Baud rate and NodeID settings remain unchanged with *RestoreDefaultParameter*.

Subindices used:

0: LargestSubindexSupported (read only)
1: RestoreAllDefaultParameters (read write)

9.1.10 Index 1014: CobIdEmergencyMessage (read write)

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

9.1.11 Index 1017: ProducerHeartbeatTime (read write)

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

9.1.12 Index 1018 IdentityObject

The Identity Object identifies the HDA 4000 CAN. The identification consists of four 32bit numbers. The combination of these 4 numbers produces a device ID which is unique worldwide.

Subindices used:

0: LargestSubIndexSupported (read only)
1: VendorID (read only)
Unique manufacturer code (0xda for HYDAC ELECTRONIC GmbH)
2: ProductCode (read only)
Hydac Electronic product code (0x0001xxxx for HDA 4000 CAN)
3: RevisionNumber (read only)
Revision number of the device
4: SerialNumber (read only)
Serial number of the device

9.1.13 Index 1800: TransmitPDOParameters

These entries determine 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 long 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. With 255 another transmission takes place if the measured value differs from the value of the last transfer by more than a set amount, or if the value exceeds or falls below the pre-set limits (see *Object Dictionary, Index* 7133, 7134 and 7135).
3. *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.

Subindices used:

0: LargestSubindexSupported (read only)
1: COBIDUsedByPDO (read write)
2: TransmissionType (read write)
5: EventTimer(read write)

9.1.14 Index 1A00: TransmitPDOMapping

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 and the number of bits of the first piece of data is stored in Subindex 1, similarly for Subindex 2.

When supplied, the HDA 4000 CAN has the following entries:

Sub index	Contents	Meaning
0	2	Two values are transferred in the *PDO.*
1	0x91300120	The first value in the *PDO* is the value of *Index 9130*, *Subindex* 01 with a width of 20h (=32 bit)
2	0x61500108	The second value in the PDO is the value of *Index 6150*, *Subindex* 01 with a width of 8 bit

Subindices used:

0: NumberOfMappedApplicationObjectsInPDO (read write)
Values 0, 1 and 2 are permitted. This means: no PDO is transmitted, or one PDO with one or two values is transmitted.

1: 1stObjectToBeMapped (read write)
On the HDA 4000 CAN the following values are permitted:
0x71300110 = Actual process value1 with 16bit width, or
0x91300120 = Actual process value1 with 32bit width.

2: 2ndObjectToBeMapped (read only)
This entry is 0x61500108 and cannot be altered.

Note: The HDA 4000 CAN carries out only a 16bit evaluation. Showing the value as a 32bit value is just to comply with the *Default-PDO-Mapping* of DS404.

9.2 Device Profile Specific Entries (DS404)

9.2.1 Index 7100: FieldValue

This entry contains an unscaled value which delivers the signal unit (see "Data flow in the HDA 4000 CAN", Page 12).

Subindices used:

0: NumberOfEntries (read only)
1: FieldValue1 (read only)

9.2.2 Index 6110: SensorType

This entry contains the sensor type. Since the HDA 4000 CAN is a pressure transmitter, it delivers the value 90 (0x5A) for *Pressure Transducer*.

Subindices used:

0: NumberOfEntries (read only)
1: SensorType1 (read only)

9.2.3 Index 6111: AutoCalibration

By inputting the character string "cali", the scaling unit is calibrated to 0 (see "Data flow in the HDA 4000 CAN", Page 12). For this for the HDA 4000 CAN must be depressurised. If the value is outside the permitted tolerance band of +/- 3% FS (Full Scale = relative to the complete measuring range), the transmission will be aborted with the Code 06090030h (see [1], SDO Abort Codes).

Subindices used:

0: NumberOfEntries (read only)
1: AutoCalibration1 (write only)

9.2.4 Index 6112: OperatingMode

Use this entry to move the input channel into a special operating mode or to switch it off. On the HDA 4000 CAN the channel cannot be switched off. Therefore the entry is read-only and always delivers the value 1=*Normal Operation*.

Subindices used:

0: NumberOfEntries (read only)
1: OperatingMode1 (read only)

9.2.5 Index 6114: ADCSampleRate

Use this entry to stipulate the sample rate. The figure is a multiple of milliseconds, according to the standard, the HDA 4000 CAN always rounds down to whole milliseconds. All figures under 1000 produce a sample rate of 1 millisecond.

Subindices used:

0: NumberOfEntries (read only)
1: ADCSampleRate (read write)

9.2.6 Index 7120: InputScaling1FV

This entry is used for scaling the process value and contains the value of the *FieldValue* for the lower measuring range limit. Normally it corresponds to a pressure of 0 bar (see "Data flow in the HDA 4000 CAN", Page 12 and [1], *Analogue Input Function Block*).

Subindices used:

0: NumberOfEntries (read only)
1: InputScaling1FV1 (read only)

9.2.7 Index 7121: InputScaling1PV

This entry is used to scale the process value and contains the value of *ProcessValue* for the lower measuring range limit. Normally it corresponds to a pressure of 0 bar. By changing this entry, any measuring range can be implemented. If, for example, the measuring range is to be 50..700 kg, then the value 50 must be input (see "Data flow in the HDA 4000 CAN", Page 12 and [1], *Analogue Input Function Block*).

Subindices used:

0: NumberOfEntries (read only)
1: InputScaling1PV1 (read write)

9.2.8 Index 7122: InputScaling2FV

This entry is used for scaling the process value and contains the value of *FieldValue* for the upper measuring range limit (see "Data flow in the HDA 4000 CAN", Page 12 and [1], *Analogue Input Function Block*).

Subindices used:

0: NumberOfEntries (read only)
1: InputScaling2FV1 (read only)

9.2.9 Index 7123: InputScaling2PV

This entry is used for scaling the process value and contains the value of *ProcessValue* for the upper measuring range limit. By changing this entry, any measuring range can be implemented. If, for example, the measuring range is to be 50..700 kg, then the value 700 must be input (see "Data flow in the HDA 4000 CAN", Page 12 and [1], *Analogue Input Function Block*).

Subindices used:

0: NumberOfEntries (read only)
1: InputScaling2PV1 (read write)

9.2.10 Index 7124: InputOffset

This entry defines an additional offset for the process value (see "Data flow in the HDA 4000 CAN", Page 12 and [1], *Analogue Input Function Block*).

Subindices used:

0: NumberOfEntries (read only)
1: InputOffset1 (read write)

9.2.11 Index 7125: InputAutoZero

By inputting the character string "zero", the *InputOffset* is set in such a way that the actual process value delivers 0 (see "Data flow in the HDA 4000 CAN", Page 12 and [1], *Analogue Input Function Block*).

Subindices used:

0: NumberOfEntries (read only)
1: InputAutoZero1 (write only)

9.2.12 Index 7130/9130: ProcessValue

These entries contain the actual process value. Index 7130 has a data width of 16bit and index 9130, 32bit. Since the HDA 4000 CAN always calculates using a data width of 16bit, both entries deliver the same value. However both entries are implemented to enable specification-compliant *PDO-Mapping*. This stipulates the 32 bit value as the default. To reduce the data traffic in the network, the 16bit value can however be *mapped* (see "Data flow in the HDA 4000 CAN", on Page 12 and "TransmitPDOMapping" on Page 20).

Subindices used:

0: NumberOfEntries (read only)
1: ProcessValue1 (read only)

9.2.13 Index 6131: PhysicalUnitProcessValue

This entry contains the unit of measurement of the process value according to the recommendation DR303.
The HDA 4000 CAN is set to "bar" when supplied. The preset unit of measurement has no effect on signal processing. To interpret the data, see [3]

Subindices used:

0: NumberOfEntries (read only)
1: PhysicalUnitProcessValue1 (read write)

9.2.14 Index 6132: ProcessValueDecimalDigits

This entry contains the number of decimal digits for interpreting the process value.

<u>Example:</u>
If the *ProcessValueDecimalDigits* is set to 2 and *PhysicalUnitProcessValue* is set to "bar", then a *ProcessValue* of 4711 corresponds to the measured value 47.11 bar.

Subindices used:

0: NumberOfEntries (read only)
1: ProcessValueDecimalDigits1 (read write)

9.2.15 Index 7133: InterruptDeltaPV

This entry is used to control the *PDO* transmission. If the actual *ProcessValue* deviates from the last one transmitted by more than the *InterruptDeltaPV*, it is transmitted. This prevents constant transmissions where the contents are very similar. In order to activate this mechanism, *TransmissionType* must be set to *ProfileSpecific* (see *TransmissionType,* Page 19).

If *InterruptDeltaPV* is set to 0, then this mechanism is also deactivated.

Subindices used:

0: NumberOfEntries (read only)
1: InterruptDeltaPV1 (read write)

9.2.16 Index 7134: InterruptLowerLimit

This entry is used to control the *PDO* transmission. If the actual *ProcessValue* falls below the *InterruptLowerLimit*, the PDO is transmitted. In order to avoid constant transmissions because the measured value is fluctuating around the limit value, the actual value must rise above the limit again by at least 1% of the preset measuring range, before another transmission is made for a fall in value.

In order for this mechanism to be activated, the *TransmissionType* must be set to *ProfileSpecific* (See *TransmissionType,* Page 19).

Subindices used:

0: NumberOfEntries (read only)
1: InterruptLowerLimit1 (read write)

9.2.17 Index 7135: InterruptUpperLimit

This entry is used to control the *PDO* transmission. If the actual *ProcessValue* exceeds the *InterruptUpperLimit*, the PDO is transmitted. In order to avoid constant transmissions because the measured value is fluctuating around the limit value, the actual value must fall below the limit again by at least 1% of the preset measuring range, before another transmission is made for this event.

In order for this mechanism to be activated, the *TransmissionType* must be set to *ProfileSpecific* (See *TransmissionType,* Page 19).

Subindices used:

0: NumberOfEntries (read only)
1: InterruptUpperLimit1 (read write)

9.2.18 Index 6150: AnlogInputStatus

This entry indicates the status of the signal unit. The status byte has the following meaning:

Bit 0: 1=fault on the signal input
Bit 1: 1=positive overload present
Bit 2: 1=negative overload present

Subindices used:

0: NumberOfEntries (read only)
1: AnlogInputStatus1 (read only)

9.2.19 Index 61A0: FilterType

This entry determines the filter type. 1 indicates *Moving average*, 2 indicates *Repeating average*. For all other values, there is no filtration. The exact procedure is described in profile [2].

Subindices used:

0: NumberOfEntries (read only)
1: FilterType1 (read write)

9.2.20 Index 61A1: FilterConstant

This entry determines the filter constants. Filtration only takes place if the constant is higher than 1. The exact procedure is described in profile [2].

Subindices used:

0: NumberOfEntries (read only)
1: FilterConstant1 (read write)

9.3 Manufacturer Specific Entries

9.3.1 Index 2001: NodeID (read write)

The *Node-ID* is stored in this index. To set the *NODE-ID* the character string "set" must be added to the address for safety.

NodeID	´s´=0x73	´e´=0x65	´t´=0x74

For example, to set the *Node-ID* to 5, the following data bytes must be transmitted in the SDO: 0x05, 0x73, 0x65, 0x74. Because of the low-high representation, with CANopen therefore, the following 32-bit value must be transmitted: 0x74657302.

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

9.3.2 Index 2002: Baud rate (read write)

The Baud rate is stored in this index. To set the Baud rate, the character string "set" must be added for safety.

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
Baud rate	1 Mbit/s	800 kbit/s	500 kbit/s	250 kbit/s	125 kbit/s	50 kbit/s	20 kbit/s	10 kbit/s

Baud rate	´s´=0x73	´e´=0x65	´t´=0x74

For example, to set the Baud rate to 125kbit/s, the following data bytes must be transmitted in the SDO: 0x04, 0x73, 0x65, 0x74. Because of the low-high representation, with CANopen therefore, the following 32-bit value must be transmitted: 0x74657304.

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

9.3.3 Index 2120: OriginalInputScaling1FV

This entry contains the original lower *FieldValue*. When supplied, the value is identical to Index 7120 *InputScaling1FV*. *InputScaling1FV* can however be changed during operation to implement other process variables. If the command *RestoreDefaultParameters* is received, *OriginalInputScaling1FV* is copied to *InputScaling1FV*.

Subindices used:

0: NumberOfEntries (read only)
1: OriginalInputScaling1FV1 (read only)

9.3.4 Index 2121: OriginalInputScaling1PV

This entry contains the original lower measuring range limit. When supplied the value is identical to Index 7121 *InputScaling1PV*. *InputScaling1PV* can however be changed during operation to implement other process variables. If the command *RestoreDefaultParameters* is received, the *OriginalInputScaling1PV* is copied to *InputScaling1PV*.

Subindices used:

2: NumberOfEntries (read only)
3: OriginalInputScaling1PV1 (read only)

9.3.5 Index 2122: OriginalInputScaling2FV

This entry contains the original upper *FieldValue*. When supplied the value is identical to Index 7122 *InputScaling2FV*. *InputScaling2FV* can however be changed during operation to implement other process variables. If the command *RestoreDefaultParameters* is received, *OriginalInputScaling2FV* is copied to *InputScaling2FV*.

Subindices used:

0: NumberOfEntries (read only)
1: OriginalInputScaling2FV1 (read only)

9.3.6 Index 2123: OriginalInputScaling2PV

This entry contains the original upper measuring range limit. When supplied the value is identical to index 7123 *InputScaling2PV*. *InputScaling2PV* can however be changed during operation, to implement other process variables. If the command *RestoreDefaultParameters* is received, the *OriginalInputScaling2PV* is copied to *InputScaling2PV*.

Subindices used:

2: NumberOfEntries (read only)
3: OriginalInputScaling2PV1 (read only)

9.3.7 Index 2131: OriginalPhysicalUnitProcessValue

This entry contains the original unit. When supplied, the value is identical to Index 6131 *PhysicalUnitProcessValue*. *PhysicalUnitProcessValue* can however be changed during operation to implement other process variables. If the command *RestoreDefaultParameters* is received, *OriginalPhysicalUnitProcessValue* is copied to *PhysicalUnitProcessValue*.

Subindices used:

0: NumberOfEntries (read only)
1: OriginalPhysicalUnitProcessValue1 (read only)

9.3.8 Index 2132: OriginalDecimalDigitsProcessValue

This entry contains the original number of decimal places. When supplied the value is identical to Index 6131 *DecimalDigitsProcessValue*. *DecimalDigitsProcessValue* can however be changed during operation to implement other process variables. If the command *RestoreDefaultParameters* is received, *OriginalDecimalDigitsProcessValue* is copied to *DecimalDigitsProcessValue*.

Subindices used:

0: NumberOfEntries (read only)
1: OriginalDecimalDigitsProcessValue1 (read only)

10 Further literature:

[1] DS301, Version 4.02
Application Layer and Communication Profile
[2] DS404, Version 1.2
Device Profile Measuring Devices and Closed-Loop Controllers
[3] DR303-2, Version 1.0
Representation of SI Units and Prefixes