CANopen for SCHUNK Modules

1. Classification

The descriptions within this document are applicable to the following mechatronical modules from SCHUNK:

- PRL-120-plus, PRL-100-plus, PRL-80-plus
- ERB-145, ERB-115
- PG-70-plus

These modules implement the CANopen communication protocol according to the recommendations of the CiA (CAN in Automation http://www.can-cia.org). Common information about CANopen can be found on the CiA homepage or here: http://en.wikipedia.org/wiki/CANopen or here: http://www7.informatik.uni-erlangen.de/~dulz/fkom/06/Material/3/CanOpen.htm

2. Standards

For the implementation in the firmware the especially the following standards and recommendations were used:

- Common CANopen protocol description:
  CiA Draft Standard DS301 version 4.2 „CANopen application layer and communication profile“
- Device profile for motion control:
  IEC 61800-7-201 „Adjustable speed electrical power drive systems –
  Part 7-201: Generic interface and use of profiles for power drive systems –
  Profile type 1 specification
  (formerly known as : CiA DS402 Device Profile for Drives and Motion Control)
- For configuration of module-ids and baud rate:
  CiA 305 DSP V2.2 CANopen layer setting services (LSS) and protocols

The standards can be bought from the IEC http://www.iec.ch/, e.g. http://webstore.iec.ch/webstore/webstore.nsf/Artnum_PK/38687. Some parts can be downloaded from the CiA web pages http://www.can-cia.org/index.php?id=specifications&no_cache=1 (eventually in an outdated version).

Currently all mandatory parts of the standards and device profile are implemented, but not all optional parts. Additionally some manufacturer specific parts are implemented in accordance to the CiA recommendations.

3. Features

3.1. Electronic Data Sheet

CANopen provides features as “***objects***”. All objects of a module are organized in an object dictionary. The objects can be accessed via a 16 bit index and an 8 bit sub index. The objects are listed in so called ***EDS files*** (Electronic Data Sheet). An EDS file describes the basic properties of the objects like name, index, sub index, data type, access rights and so on. EDS files come in two flavors:

- The older *.eds format according to CiA 306
- The newer XML based *.xdd format according to ISO 15745-2

For SCHUNK modules the EDS files are provided in both formats.

At https://www.vector.com/canopen_caneds_en.html a freeware tool to view (and even edit) EDS files (in both formats) is available.

3.2. Object description

A detailed description of the underlying *meaning* of a CANopen object is not contained in an EDS file. The ***common*** CANopen objects as well as the ***device profile specific*** objects are verbosely described in the official standard documents mentioned in section 2. That information will not be repeated here, except for the examples in section 3.2.2.

3.2.1. Manufacturer specific objects

The ***manufacturer specific*** objects as implemented in the SCHUNK CANopen firmware will be described in the following sections. For the basic properties of the objects like data type access rights please refer to the EDS file. Here only the name, index, subindex will be mentioned along with a verbose description of the meaning.

The subsection headings below will be in the format *index/subindex name*

○ 0x2008/0 Password

Some of the objects can only be overwritten with increased user access rights. In order to gain these user access rights the correct password has to be written via SDO into the ***Password*** object. The following passwords are available for end users:

- Access level ‚diagnosis‘, password: Diag
- Access level ‚profi‘, password: Schunk
- Access level ‚advanced‘, password: ?SCHUNK!

In case that an invalid password is given the user access level will be reset to “guest”.

○ 0x2009/0 User

Numeric representation of the currently active user access rights as set via 0x2008/0 Password. The value can be read via SDO. The following users/numeric values are available:

- 0 Guest (default level)
- 1 Diagnosis
- 2 Profi
- 3 Advanced

○ 0x200a/0 Disconnect_Reset

For very specific circumstances it might be necessary to perform one of the operations described below with a module. In order to trigger the operation the numeric value must be written to the object

- 1 Disconnect: The module will disconnect from the CAN bus and will not respond to further requests. To access the module again the logic supply voltage must be switched off and on again.
- 2 Reset: The module will perform a reset and start over again

The following operations are available with firmware build date of 2012-10-22 and later only:

- 3 Dereference and reset: The module will lose its referenced state and reset. If the module was referenced before then it will no longer be referenced immediately after the reset
- 4 Dereference and reset: The module will lose its referenced state on the first movement after reset and reset. If the module was referenced before then it will still be referenced immediately after the reset, but on the first movement the referenced state will be lost when the first index pulse is seen.

○ 0x200b error_details

Information about the details of the error that occurred latest can be read from the subindices.

○ 0x200b/0 Number of entries

As usual for objects with sub indices other than zero the entry with sub index 0 provides the highest sub index used. Here “3” can be read.

○ 0x200b/1 detail

A numeric float value describing further details about an occurred error. When reporting errors back to SCHUNK please include the value of this ***detail*** object.

○ 0x200b/2 file

A string describing the source code file name of the error location. When reporting errors back to SCHUNK please include the value of this ***file*** object.

○ 0x200b/3 line

A numeric integer value describing the source code line number of the error location. When reporting errors back to SCHUNK please include the value of this ***line*** object.

- **0x200c/0 communication_mode**

Numeric value for the communication mode to use. Warning: If something other than “5” = CANopen is written to communication_mode then communication via the CANopen protocol will no longer be available after restart. The following modes/numeric values are available:

- 1 RS232 or USB
- 2 CAN
- 5 CANopen

- **0x200d pcb_temperature**

Information about the PCB (Printed Circuit Board) temperature can be read/written from/to the subindices

- **0x200d/0 Number of entries**

As usual for objects with sub indices other than zero the entry with sub index 0 provides the highest sub index used. Here “2” can be read.

- **0x200d/1 actual_pcb_temperature**

Numeric float value of the last measured PCB temperature in degrees Celsius. This object can be mapped to a PDO.

- **0x200d/0 pcb_temperature_update_period_ms**

Update period of the PCB temperature measurement in milliseconds. The default value is 20000 ms = 20 s. Values smaller than 1000 (1s) and greater than 3600000 (1h) are not allowed. If you query the ***actual_pcb_temperature*** more often than ***pcb_temperature_update_period_ms*** then you will read the same value multiple times.

- **0x200e/0 sync_timeout_factor**

Numeric factor which indicates how many SYNC periods can be missed before the module performs a fast stop. The default value is 6. This value is used in ‘Interpolated position mode’ only.

This parameter can be useful while starting up a system or while testing/debugging/experimenting. Smaller values will make the module stop earlier if the superior robot control system stops sending SYNC commands. Larger values will make the module stop later if the superior robot control system stops sending SYNC commands. A value of 0 will disable the timeout, i.e. even if no further SYNC commands are received the module will stay in control.

While testing / debugging the superior robot control system it might be useful to increase the value of ***sync_timeout_factor*** or to disable it altogether by setting it to zero. This will keep the module going even with larger time jitter of the SYNC commands.

- **0x2010/0 KR_Current**

Proportional parameter of the current controller of the axis. This value is accessible with user access level ‘profi’ or higher (see 0x2008/0 Password). Please adjust this parameter only when instructed by SCHUNK employees.

- **0x2011/0 TN_Current**

Reset Time parameter of the current controller of the axis. This value is accessible with user access level ‘profi’ or higher (see 0x2008/0 Password). Please adjust this parameter only when instructed by SCHUNK employees.

- **0x2012/0 TD_Current**

Differential parameter of the current controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). Value should be zero.

- **0x2013/0 KC_Current**

Anti windup parameter of the current controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). The value should be one.

- **0x2020/0 KR_Speed**

Proportional parameter of the speed controller of the axis. This value is accessible with user access level ‘profi’ or higher (see 0x2008/0 Password). Please adjust this parameter only when instructed by SCHUNK employees.

- **0x2021/0 TN_Speed**

Reset time parameter of the speed controller of the axis. This value is accessible with user access level ‘profi’ or higher (see 0x2008/0 Password). Please adjust this parameter only when instructed by SCHUNK employees.

- **0x2022/0 TD_Speed**

Differential parameter of the speed controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). Value should be zero.

- **0x2023/0 KC_Speed**

Anti windup parameter of the speed controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). The value should be one.

- **0x2024/0 KS_FeedForward**

Current feed forward parameter of the speed controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). Value should be zero.

- **0x2030/0 KR_Position**

Proportional parameter of the position controller of the axis. This value is accessible with user access level ‘profi’ or higher (see 0x2008/0 Password). Please adjust this parameter only when instructed by SCHUNK employees.

- **0x2031/0 TN_Position**

Reset time parameter of the position controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). Value should be zero.

- **0x2032/0 TD_Position**

Differential parameter of the position controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). Value should be zero.

- **0x2033/0 KC_Position**

Anti windup parameter of the Position controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). The value should be one.

- **0x2034/0 KP_FeedForward**

Speed feed forward parameter of the position controller of the axis. This value is not accessible with public user access levels (see 0x2008/0 Password). Value should be one.

- **0x2041/0 Delta_Position**

Position window in millidegrees / µm for reporting target reached. This value is accessible with user access level ‘profi’ or higher (see 0x2008/0 Password). Please adjust this parameter only when instructed by SCHUNK employees.

- **0x2050/0 extended_status**

Bit vector with additional status information. This object is available with firmware build date of 2012-10-22 and later only. The object can be mapped to a PDO

- Bit 0 (Bit mask 0x01) flag: commutation_search_completed
- Bit 1 (Bit mask 0x02) flag: pseudo_absolute_position_verified

The interpretation of these flags needs a little more verbose explanation:

Background “commutation search”:

The SCHUNK modules use EC motors (electronically commutated) with sinus commutation. This requires to determine a so called sinus vector to be able to drive the motor.

For the PRL-plus and PG-70-plus modules the sinus vector is determined once. But the known sinus vector has to be verified once after each logic power on. This is done by starting the first movement after logic power on with block commutation.

The ERB modules, which do not have hall sensors needed for block commutation, will have to search the sinus vector once after each logic power on. This is done automatically by performing small (few degrees) movements right before the first movement command is performed.

Extension ***commutation_search_completed***:
In order to be able to query the state of the sinus vector verification or the commutation search the ***extended_status*** was introduced. The ***commutation_search_completed*** flag yields 0 as long as the verification / search has not (yet) been finished and 1 once it has been finished.

Background "pseudo absolute position":
The SCHUNK modules do not contain a physical absolute position sensor. But the firmware implements a so called pseudo absolute encoder which makes the modul keep its referenced state even if after a power cycle under suitable circumstances. To make this work the logic power must be switched off only while the module is not in control (brakes applied) and the axis must not be moved while the logic power is off. The latter is verified on the first movement which might lead to a delayed loss of the referenced state.

The actual "referenced bit" (bit 15, bit-mask 0x8000) in the 'normal' statusword (object 0x6041/0) is reported as set (1) after start (logic power on), if the robot was referenced and it was shut down in the 'normal' fashion before, i.e. the brakes were applied on logic power off. But the referenced state can be lost during the first movement after power on. In case that the axis of the module has moved while switched off (e.g. due to vibrations or excessive external force), then the robot is in fact no longer referenced. But the robot cannot determine that right away and thus reports "referenced" to begin with. The verification of the referenced state can only be done when passing an encoder index pulse. In case the measured distance to the index pulse differs from the expected one then the referenced state will be unset and the referenced bit will be cleared in the statusword.

Consequences:
It is not sufficient to monitor the referenced bit only once after start. Instead the referenced state must be checked after the axis has moved a certain minimum distance in one direction. The minimum distance depends on the gear ratio:

- ERB145 0.746 deg = 746 mdeg
- ERB1155 1.200 deg = 1200 mdeg
- PRL-80-plus 0.653 deg = 653 mdeg
- PRL-100-plus 0.576 deg = 576 mdeg
- PRL-120-plus 0.605 deg = 605 mdeg
- PG-70-plus 1.000 mm = 1000 µm

Extension ***pseudo_absolute_position_verified***:
In order to be able to query the state of the verification of the pseudo absolute position the ***extended_status*** was introduced. The ***pseudo_absolute_position_verified*** flag yields 0 as long as the pseudo absolute position has not (yet) been verified and 1 if it has been verified.

Further consequences for the ERB modules:
During the sinus vector search on the the first movement an encoder index pulse can be seen, but it doesn't have to be. Depending on the starting position a smaller movement can be sufficient for searching for the sinus vector. In consequence the state information ***commutation_search_completed*** and ***pseudo_absolute_position_verified*** are independent of each other

3.2.2. Device profile specific objects

The supported device profile specific objects as listed in the EDS files work as described in the standards.

Only the write access for the following two objects 0x607d/1 and 0x607d/2 is restricted, see object 0x2008/0.

- **0x607d/1 software_position_limit_min_position_limit**

Minimum axis position in milli degrees or µm. This is also known as the soft limit low. This value is accessible with user access level 'profi' or higher (see 0x2008/0 Password).

- **0x607d/2 software_position_limit_max_position_limit**

Maximum axis position in milli degrees or µm. This is also known as the soft limit high. This value is accessible with user access level 'profi' or higher (see 0x2008/0 Password).

- **0x60fe/1 digital_outputs_physical_outputs**

Value of the digital outputs of the module. Currently none of the SCHUNK CANopen modules supports digital outputs. But the object is implemented nonetheless since it provides access to the brake of the module. You can release the brake by setting the lowest bit of 0x60fe/1 to 0. To apply the brake again set the lowest bit to 1.

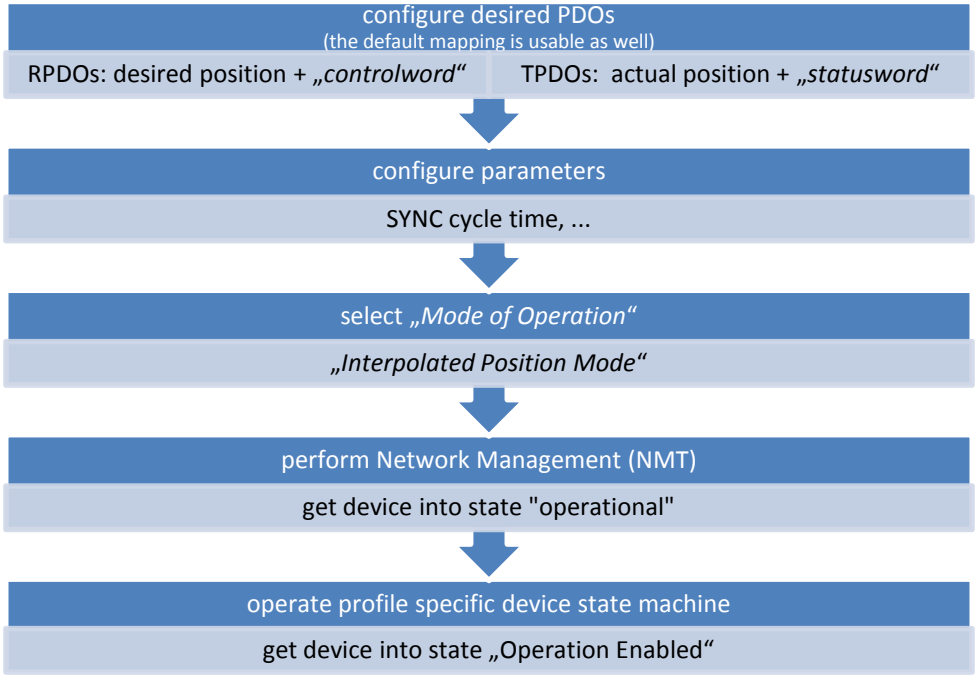
This direct brake control is available only if:

- The NMT (Network Management) state is “operational”
- The voltage is not applied to the motor, i.e. normal controller is not actively controlling the motor

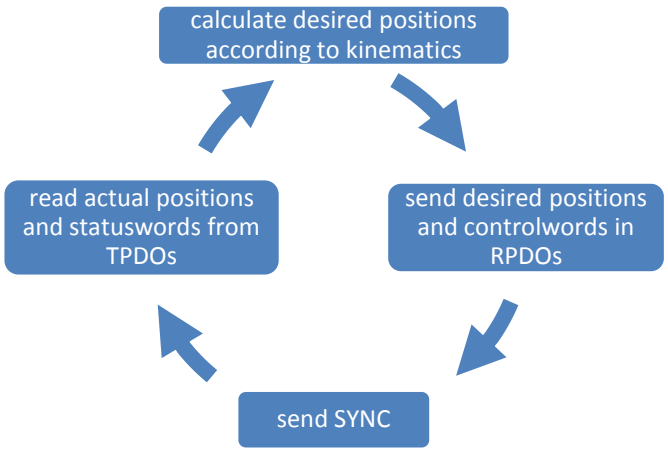
4. Robot control

The „interpolated position mode“ is specifically designed for the coordinated control of multi axis system in the sense of a robot control system. The mode is described in detail in the older “*Draft Standard“* DS402 in chapter 15 and in the newer IEC 61800-7-201 standard in chapter 13. A very coarse overview is shown below.

General approach for using a CANopen-Module in „Interpolated Position Mode“ from the viewpoint of the superior robot control system:



Afterwards the following operations are performed cyclically:



5. Error Codes

The SCHUNK CANopen firmware supports the error codes as specified in the common CiA Draft Standard DS301 section 7.2.7.1 page 69 as well as the device profile specific error codes from IEC 61800-7-201 section 7.1 page 35. Not all error codes from the standards are used. Those that are used are listed below.

5.1. Error Codes used by SCHUNK CANopen firmware

5.1.1. CANopen DS301

- **0x0000 Error reset or no error**
- **0x1000 Generic error**
- **0x6100 Internal software – generic**
- **0x8100 Communication – generic**
- **0x8110 CAN overrun (objects lost)**

The module could not process incoming CAN data as quickly as required, therefore communication objects have been lost.

- **0x8210 PDO not processed due to length error**
- **0x8250 RPDO timeout**

5.1.2. Device profile CANopen DS402

- **0x2110 Short circuit/earth leakage (input)**
- **0x2310 Continuous over current**

A motor current that is too high has been applied for too long.

- **0x2350 Load level fault (I2t, thermal state)**

The module monitors the motor load level using a simplified motor temperature model that uses the motor current (I) and the time (t). If currents above the nominal current have been applied for too long this I2t error is signaled to protect the motor from overheating.

- **0x3211 Over-voltage no. 1**

The logic voltage is too high

- **0x3212 Over voltage no. 2**

The motor voltage is too high

- **0x3221 Under-voltage no. 1**

The logic voltage is too low

- **0x3222 Under-voltage no. 2**

The motor voltage is too low

- **0x4210 Excess temperature device**
- **0x4220 Too low temperature device**
- **0x4310 Excess temperature drive**
- **0x4410 Excess temperature supply**
- **0x5520 ROM/EPROM**
- **0x5530 EEPROM**
- **0x6320 Parameter error**

This error is reported fort the following internal error conditions:

- INFO_POSITION_NOT_REACHABLE: commanded position is beyond software limits
- INFO_WRONG_PARAMETER: parameter is wrong
- ERROR_WRONG_RAMP_TYPE: parameter type is wrong
- ERROR_INITIALIZE: error during initialization, check EEPROM parameters.

- **0x7122 Motor error or commutation malfunc**

Commutation error, sinus vector could not be found. Possible causes:

- Position measurement system defective
- Movement of axis is mechanically blocked
- Current sensor defective or not calibrated
- Motor phase defective

- **0x7303 Resolver 1 fault**

- **0x7305 Incremental sensor 1 fault**

Module could not be referenced due to an error with the incremental position sensor

- **0x7320 Position**

- **0x8400 Velocity speed controller**

The measured velocity has been too high. For safety reasons the module was stopped and the error is reported.

- **0x8611 Following error**

Actual position of the module could not follow the commanded target position.

- **0x8612 Reference limit**

Actual position of the module is out of the specified limits. After acknowledging the error only movements towards the allowed movement range are allowed.

- **0x8900 Process data monitoring**

- **0xFF00 <MANUFACTURER SPECIFIC CODES>**

For completeness only. It is very unlikely that this error code is reported. The internal, manufacturer specific errors behind this error codes include:

INFO_BATTERY_LOW, ERROR_BATTERY_DOWN, INFO_PROGRAM_END, ERROR_PROGRAM_ABORTION, ERROR_PROGRAM_RUNTIME, ERROR_PROGRAM_INVALID, ERROR_PROGRAM_CONFLICT, ERROR_UNEXPECTED_STOP_DURING_PHRASE, ERROR_PRESSURE, ERROR_SERVICE, ERROR_VPC3, ERROR_ANALOG_INPUT

- **0xFF01 ERROR_HARDWARE_VERSION**

- **0xFF02 ERROR_MATH**

- **0xFF02 INFO_NO_RIGHTS**

A parameter write access has been refused due to insufficient rights. See also the manufacturer specific parameter 0x2008/0.

- **0xFF01 INFO_UNKNOWN_COMMAND**

- **0xFF02 INFO_FAILED**