# ProNet CANOpen User's Manual

（Version：V1.04）

ESTUN AUTOMATION TECHNOLOGY CO., LTD

— Total Solution Supplier

# ——Contents ——

# Chapter 1 Brief introduction

## 1.1 CAN main files

| Document Name | Source |
|---|---|
| **CiA DS 301 V 4.01:**<br>CANopen Communication Profile<br>for Industrial Systems - based on CAL | **CiA** |
| **CiA DSP 402 V 2.0:**<br>CANopen Device Profile | **CiA** |

## 1.2 Terms and abbreviations

| | |
|---|---|
| **CAN** | Controller Area Network |
| **CiA** | CAN in Automation International Users and Manufacturers Group. |
| **COB** | Communication Object (CAN message). A unit of transportation in a CAN network. Data must be sent across a network inside a COB. The COB itself is part of the CAN message frame. |
| **EDS** | Electronic Data Sheet. A node-specific ASCII-format file required when configuring the CAN network.The EDS file contains general information on the node and its dictionary objects (parameters). |
| **LMT** | Layer Management. One of the service elements of the CAN Application Layer in the CAN Reference Model. It serves to configure parameters for each layer in the CAN Reference Model. |
| **NMT** | Network Management. One of the service elements of the CAN Application Layer in the CAN Reference Model. It performs initialization, configuration and error handling on a CAN network. |
| **OD** | A local storage of all Communication Objects (COB) recognized by a device. |
| **Parameter** | A parameter is an operating instruction for the drive. Parameters can be read and programmed with the drive control panel. |
| **PDO** | Process Data Object; a type of COB. Used for transmitting time-critical data, such as control commands, references and actual values. |
| **RO** | Denotes read-only access. |
| **RW** | Denotes read/write access. |
| **SDO** | Service Data Object; a type of COB. Used for transmitting non-time critical data, such as parameters. |

# 1.3 Brief introduction

CANopen is a higher-layer protocol based on the CAN (Control Area Network) serial bus system and the CAL (CAN Application Layer). CANopen assumes that the hardware of the connected device has a CAN transceiver and a CAN controller as specified in ISO 11898.

The CANopen Communication Profile, CiA DS-301, includes both cyclic and event-driven communication, which makes it possible to reduce the bus load to minimum while still maintaining extremely short reaction times. High communication performance can be achieved at relatively low baud rates, thus reducing EMC problems and cable costs.

CANopen device profiles define both direct access to drive parameter and time-critical process data communication. The NCAN-02 fulfils CiA (CAN in Automation) standard DSP-402 (Drives and Motion Control), supporting the 'Manufacturer Specific' operating mode only.

The physical medium of CANopen is a differentially-driven two-wire bus line with common return according to ISO 11898. The maximum length of the bus is limited by the communication speed as follows:

| Baud Rate | Max. Bus Length |
|---|---|
| 1M bit/s | 25 m |
| 500k bit/s | 100 m |
| 250k bit/s | 250 m |
| 125k bit/s | 500 m |
| 100k bit/s | 600 m |
| 50k bit/s | 1000 m |

The maximum theoretical number of nodes is 127. However, in practice, the maximum number depends on the capabilities of the CAN transceivers used.

Further information can be obtained from the CAN in Automation International Users and Manufacturers Group (*www.can-cia.de*).

# Chapter 2 Cabling and wiring

**●The layout of CN3 terminal**

| Pin number | Name | Function |
|---|---|---|
| 1 | — | Reserved |
| 2 | — | |
| 3 | 485+ | RS-485 communication terminal |
| 4 | ISO_GND | Isolated GND |
| 5 | ISO_GND | |
| 6 | 485- | RS-485 communication terminal |
| 7 | CANH | CAN communication terminal |
| 8 | CANL | CAN communication terminal |

Note: Do not short terminal 1 and 2 of CN3

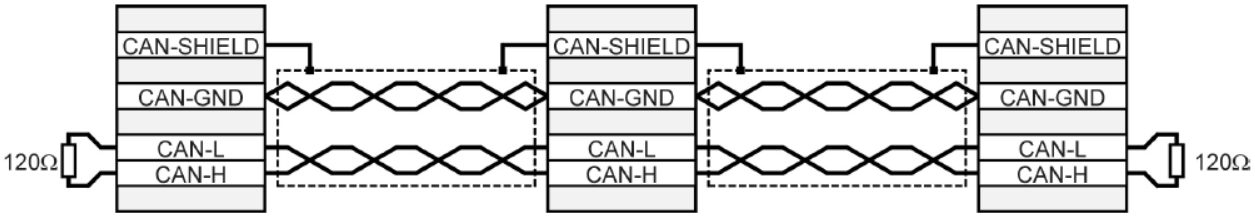
**● The layout of CN4 terminal**

| Pin number | Name | Function |
|---|---|---|
| 1 | — | Reserved |
| 2 | — | |
| 3 | 485+ | RS-485 communication terminal |
| 4 | ISO_GND | Isolated GND |
| 5 | ISO_GND | |
| 6 | 485- | RS-485 communication terminal |
| 7 | CANH | CAN communication terminal |
| 8 | CANL | CAN communication terminal |

CN3 is always the input terminal of communication cable and CN4 is always the output terminal of communication cable. (If connection to another communication node is necessary, the cable will connect CN4 to next communication node. If not, a terminal resistor could be applied at CN4). When multiple ProNet devices are connected, it is forbidden to connect the CN3 terminals of different drives directly.

For example, a network is composed of one PLC, three ProNet drives called A, B and C. The cabling network is as below: PLC → CN3 of drive A，CN4 of drive A → CN3 of drive B，CN4 of drive B → CN3 of drive C, CN4 of drive C → 120Ω resistor.

The two ends of the CAN cable have to be terminated by a resistor of 120Ω (1%, 1/4W) as below.




Please select the bus cable with double twisted pair cables and shielding layer, one pair for connecting CAN-L and CAN-H, another pair for grounding.

# Chapter 3 CANopen communication

CAL supplies all network management service and message transferring protocol with defining the content of object or type of object for communication. It defines how instead of what, which is the strength of CANopen.

CANopen is developed based on CAL. It applies CAL protocol subsets for communication and service and creates a solution to DCS. CANopen could freely extend the node function to simplicity or complex while the network nodes are accessible and available to each other.

The key concept of CANopen is object dictionary. This way of object description is also applied to other fieldbus system like Probis and Interbus-S. CANopen communication could access to all the parameter of drivers through object dictionary. Please notice object dictionary is not one part of CAL, instead of which it is realized in CANopen.

CANopen communication defines several types of objects as below…

| Abbreviation | Full Spell | Description |
|---|---|---|
| SDO | Service Data Object | Used for normal parameterization of the servo controller |
| PDO | Process Data Object | Fast exchange of process data (e.g. velocity actual value) possible. |
| SYNC | Synchronization Message | Synchronization of several CAN nodes |
| EMCY | Emergency Message | Used to transmit error messages of the servo controller. |
| NMT | Network Management | Used for network services. For example usercan act on all controllers at the same time via this object type. |
| Heartbeat | Error Control Protocol | Used for observing all nodes by cyclic messages. |

CAN employs data frames for transferring data between the host (controller) and the nodes on the bus. The following figure presents the structure of the data frame.

| Start of frame | Arbitration field | | Control field | Date field | Cyclical redundancy check | Acknowledged field | End of frame |
|---|---|---|---|---|---|---|---|
| | COB-ID | RTR | | | | | |
| 1BIT | 1 OR 29 BITS | 1BIT | 6BITS | 0～8BYTES | 16BITS | 2BITS | 7BITS |

Our drivers doesn't support remote frame currently. The detail of COB-ID is as below.

| Function code | | | | Node ID | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

## 3.1 CAN identifier list

| Object | COB-ID bit10~7 （binary） | COB-ID (hex) | Index in OD |
|---|---|---|---|
| NMT | 0000 | $000_h$ | — |
| SYNC | 0001 | $080_h$ | $1005_h$、$1006_h$、$1007_h$ |
| TIME STAMP | 0010 | $100_h$ | $1012_h$、$1013_h$ |

| | | | |
|---|---|---|---|
| EMCY | 0001 | $081_h$ ~ $0FF_h$ | $1024_h$、$1015_h$ |
| PDO1（transimit） | 0011 | $181_h$ ~ $1FF_h$ | $1800_h$ |
| PDO1（receive） | 0100 | $201_h$ ~ $27F_h$ | $1400_h$ |
| PDO2（transimit） | 0101 | $281_h$ ~ $2FF_h$ | $1801_h$ |
| PDO2（receive） | 0110 | $301_h$ ~ $37F_h$ | $1401_h$ |
| PDO3（transimit） | 0111 | $381_h$ ~ $3FF_h$ | $1802_h$ |
| PDO3（receive） | 1000 | $401_h$ ~ $47F_h$ | $1402_h$ |
| PDO4（transimit） | 1001 | $481_h$ ~ $4FF_h$ | $1803_h$ |
| PDO4（receive） | 1010 | $501_h$ ~ $57F_h$ | $1403_h$ |
| SDO（transimit） | 1011 | $581_h$ ~ $5FF_h$ | $1200_h$ |
| SDO（receive） | 1100 | $601_h$ ~ $67F_h$ | $1200_h$ |
| Heartbeat | 1110 | $701_h$ ~ $77F_h$ | $1016_h$、$1017_h$ |

Note:

1. PDO/SDO‘s send/receive is observed by (slave) CAN.
2. Our drive‘s CANopen protocol currently supports 4 transimit PDO and 4 receive PDO.

# 3.2 SDO

SDO is used to visit the object dictionary of a device. Visitor is called client. The CANopen device whose object dictionary is visited and required to supply the asked service is called server. CANopen messages from a client and servo all contain 8 bits (Not all of them are meaningful). A request from a client must be confirmed by a server

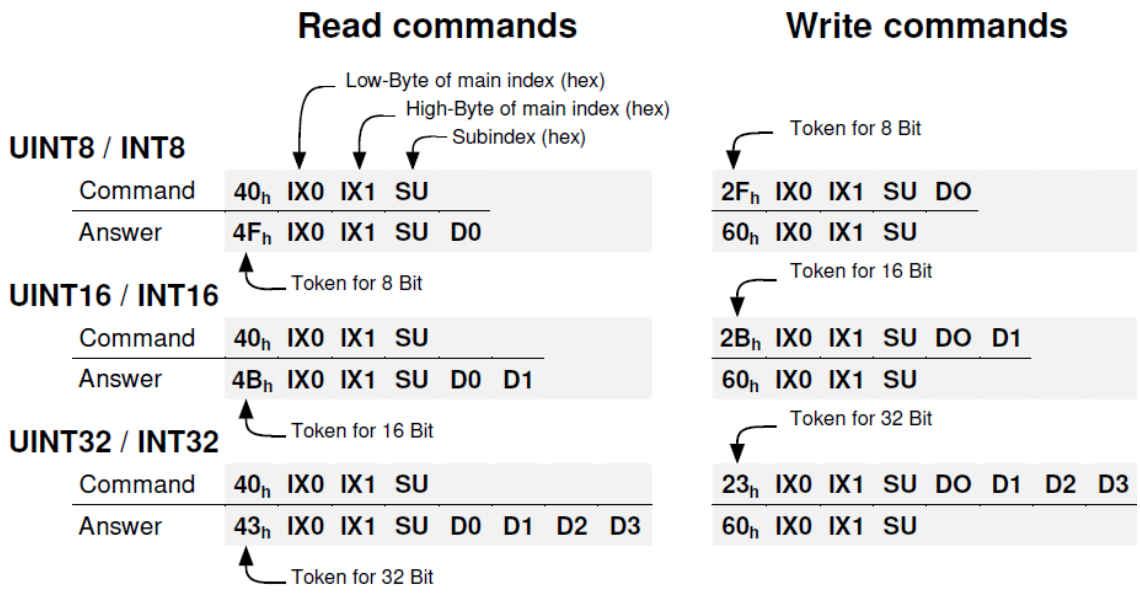
There are 2 method of conveying SDO:

- Expedited transfer：contains 4 bytes at maximum
- Segmented transfer：contains more than 4 bytes

Basic structure of SDO：

| Byte0 | Byte1～2 | Byte3 | Byte4～7 |
|---|---|---|---|
| SDO | Object reference | Sub-object reference | data |

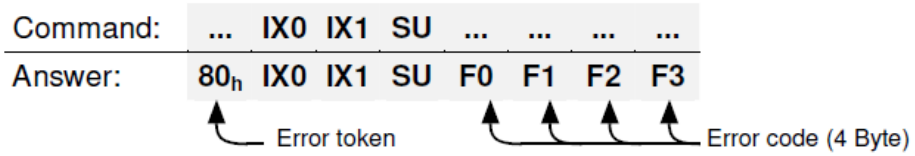
SDO read/write command structure:




Example:

| | Read | Write |
|---|---|---|
| UINT8 / INT8 | Reading of Obj. 6061_00h<br>Returning data: 01h | Writing of Obj. 1401_02h<br>Data: EFh |
| Command: | 40h 61h 60h 00h | 2Fh 01h 14h 02h EFh |
| Answer: | 4Fh 61h 60h 00h 01h | 60h 01h 14h 02h |
| UINT16 / INT16 | Reading of Obj. 6041_00h<br>Returning data: 1234h | Writing of Obj. 6040_00h<br>Data: 03E8h |
| Command: | 40h 41h 60h 00h | 2Bh 40h 60h 00h E8h 03h |
| Answer: | 4Bh 41h 60h 00h 34h 12h | 60h 40h 60h 00h |
| UINT32 / INT32 | Reading of Obj. 6093_01h<br>Returning data: 12345678h | Writing of Obj. 6093_01h<br>Data: 12345678h |
| Command: | 40h 93h 60h 01h | 23h 93h 60h 01h 78h 56h 34h 12h |
| Answer: | 43h 93h 60h 01h 78h 56h 34h 12h | 60h 93h 60h 01h |

SDO-error messages：




| Error code F3 F2 F1 F0 | Description |
|---|---|
| 05 03 00 00$_h$ | Toggle bit not alternated |
| 05 04 00 01$_h$ | Client / server command specifier not valid or unknown |
| 06 01 00 00$_h$ | Unsupported access to an object |
| 06 01 00 01$_h$ | Attempt to read a write only object |
| 06 01 00 02$_h$ | Attempt to write a read only object |
| 06 02 00 00$_h$ | Object does not exist in the object dictionary |
| 06 04 00 41$_h$ | Object cannot be mapped to the PDO |
| 06 04 00 42$_h$ | The number and length of the objects to be mapped would exceed PDO length |
| 06 04 00 47$_h$ | General internal incompatibility in the device |
| 06 07 00 10$_h$ | Data type does not match, length of service parameter does not match |
| 06 07 00 12$_h$ | Data type does not match, length of service parameter too high |
| 06 07 00 13$_h$ | Data type does not match, length of service parameter too low |
| 06 09 00 11$_h$ | Sub-index does not exist |
| 06 04 00 43$_h$ | General parameter incompatibility |
| 06 06 00 00$_h$ | Access failed due to an hardware error *1) |
| 06 09 00 30$_h$ | Value range of parameter exceeded |
| 06 09 00 31$_h$ | Value of parameter written too high |
| 06 09 00 32$_h$ | Value of parameter written too low |
| 06 09 00 36$_h$ | Maximum value is less than minimum value |
| 08 00 00 20$_h$ | Data cannot be transferred or stored to the application *1) |
| 08 00 00 21$_h$ | Data cannot be transferred or stored to the application because of local control |
| 08 00 00 22$_h$ | Data cannot be transferred or stored to the application because of the present device state *3) |
| 08 00 00 23$_h$ | No Object Dictionary is present *2) |

# 3.3 PDO

PDO is applied to transferring real time data which will be conveyed from a producer to one or multiple clients. Data transferring will be limited to 1 to 8 bytes. There is no hand-shake restriction in PDO communication, which means data has been redefined, so clients could process the received data for vary short time. PDO content will be only defined by its CAN ID, assuming producers and clients know PDO content from its CAN ID.
2 objects in object dictionary are used for each PDO.

- PDO communication parameter：It contains COB-ID, transferring type, restriction time and cycle of timer used by PDO.

- PDO mapping parameter：It contains a list of objects in the object dictionary. These objects are mapped into PDO, includes their data length in bits. Producers and clients must know this mapping to explain the content of PDO.

The content of PDO's message is predefined or configured when the network initializes. Mapping application object into PDO is described in object dictionary. If a device (producer and client) support dynamic mapping, SDO could be used to configure PDO's mapping parameter. Our servo drive supports dynamic PDO mapping. There are 2 rules for PDO mapping to follow..

1. Each PDO could be mapped into 4 objects.
2. The length of each PDO will be no more than 64 bits.

PDO mapping process：

1. Set the sub-index of PDO coordinated mapping parameter ($1600_{h, 1601\,h}$, $1A00_h$ or $1A01_h$) as o.
2. Revise the sub-index from 1 to 4 of PDO coordinated mapping parameter ($1600_{h, 1601\,h}$, $1A00_h$ or $1A01_h$).
3. Set the sub-index 0 of PDO coordinated mapping parameter($1600_h$ ,$1601_h$,$1A00_h$ or $1A01_h$ ) as legal number( number of PDO's mapping objects)
4. PDO mapping completing.

There are multiple ways to transmit PDO:

- Synchronous（Synchronization by receiving SYNC object）

  Cycle: Transmit triggered after every 1 to 240 SYNC messages.

- Asynchronous

  Transmit triggered by special object event regulated in sub-object protocol.

Transmit type of PDO

| **Transmit Type** | **Description** | **PDO** |
|---|---|---|
| 0 | Reserved | — |
| 1~240 | **SYNC**：It represents the number of SYNC objects between 2 PDOs. | TPDO/RPDO |
| 240~253 | Reserved | — |
| 254 | **Asynchronous**：If the content of PDO has changed, PDO transmit will be triggered. | TPDO |
| 255 | **Asynchronous**：<br>The content of PDO will be periodically updated and transmitted. | TPDO/RPDO |

One PDO could set a frozen time which is the shortest interval time between 2 continuous PDO. It could prevent the bus

from being occupied by amount of data with high priority. Frozen time is defined by 16 bit unsigned integer number and its unit is 100us.

One PDO could set a timing period. When the regulated time is violated, a PDO transmit could be triggered without a trigger bit. Object timing period is defined as 16 bit unsigned integer and its unit is 1ms.

PDO mapping case：

Map the 3 objects to PDO1 (transmit). PDO1 (transmit) is required to be asynchronous periodic type with period time as much as 10ms and frozen time as much as 2ms.

| Object | Index — Sub-index | Description |
|---|---|---|
| statusword | $6041_h$ – 00 $_h$ | Status word |
| modes_of_operation_display | $6061_h$ – 00 $_h$ | Practical operational mode |
| Position_Acture_Value | $6064_h$ – 00 $_h$ | Practical position |

1）Clear number_of_mapped_objects

number_of_mapped_objects(1A00 $_h$：00 $_h$)= 0

2）Set the parameter for mapping objects

Index =6041 $_h$ Subin. = 00h Length = 10 $_h$ ⇒ 1st_mapped_object(1A00 $_h$：01 $_h$)= 60410010 $_h$

Index =6061 $_h$ Subin. = 00h Length = 08 $_h$ ⇒ 2st_mapped_object(1A00 $_h$：02 $_h$)= 60610008 $_h$

Index =60FD $_h$ Subin. = 00h Length = 20 $_h$ ⇒ 3st_mapped_object(1A00 $_h$：03 $_h$) = 60FD0020 $_h$

3）Set number_of_mapped_objects

number_of_mapped_objects(1A00 $_h$：00 $_h$)= 3

4）Set PDO communication parameter

PDO1（transmit）is asynchronous periodical type ⇒ transmission_type (1800 $_h$：02 $_h$)= FF $_h$

Frozen time 2ms(20×100us) ⇒ inhibit_time (10A0 $_h$：03 $_h$)= 14 $_h$

Period time: 10ms(10×1ms) ⇒ event_time (1800 $_h$：05 $_h$)= 0A $_h$

5）PDO mapping complete.

**PDO parameter**

ProNet drive contains 4 transmit PDOs and 4 receive PDOs. The detailed communication parameter and mapping parameter of the first transmit/receive PDO is as below and those of the rest 3 transmit/receive PDO are the same as the first PDO.

| | |
|---|---|
| Index | 1800 h |
| Name | transmit_pdo_parameter_tpdo1 |
| Object Code | RECORD |
| No. of Elements | 4 |

| | |
|---|---|
| Sub-Index | 01 h |
| Description | cob_id_used_by_pdo_tpdo1 |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | 181 h...1FF h, Bit 31 may be set |
| Default Value | 181 h |

| | |
|---|---|
| Sub-Index | 02 h |
| Description | transmission_type_tpdo1 |
| Data Type | UINT8 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | 1...240,254,255 |
| Default Value | 255 |

| | |
|---|---|
| Sub-Index | 03 h |
| Description | inhibit_time_tpdo1 |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | 100μs |
| Value Range | —— |
| Default Value | 100 |

| Sub-Index | 05 h |
|---|---|
| Description | event_time_tpdo1 |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | 1ms |
| Value Range | —— |
| Default Value | 10 |

| Index | 1A00 h |
|---|---|
| Name | transmit_pdo_mapping_tpdo1 |
| Object Code | RECORD |
| No. of Elements | 2 |

| Sub-Index | 00 h |
|---|---|
| Description | number_of_mapped_objects_tpdo1 |
| Data Type | UINT8 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | 0...4 |
| Default Value | 2 |

| Sub-Index | 01 h |
|---|---|
| Description | first_mapped_object_tpdo1 |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | See table |

| Sub-Index | 02 h |
|---|---|
| Description | second_mapped_object_tpdo1 |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |

| | |
|---|---|
| Default Value | See table |

| | |
|---|---|
| Sub-Index | 03 h |
| Description | third_mapped_object_tpdo1 |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | See table |

| | |
|---|---|
| Sub-Index | 04 h |
| Description | fourth_mapped_object_tpdo1 |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | See table |

1、T-PDO1

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1800 h _00 h | number of entries | UINT8 | RO | 04 h |
| 1800 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000181 h |
| 1800 h _02 h | transmission type | UINT8 | RW | FF h |
| 1800 h _03 h | inhibit time (100 μs) | UINT16 | RW | 64 h |
| 1800 h _05 h | event time (1ms) | UINT16 | RW | 0A h |
| 1A00 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1A00 h _01 h | first mapped object | UINT32 | RW | 60410010 h |
| 1A00 h _02 h | second mapped object | UINT32 | RW | 60640020 h |
| 1A00 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1A00 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

2、T-PDO2

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1801 h _00 h | number of entries | UINT8 | RO | 04 h |
| 1801 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000281 h |
| 1801 h _02 h | transmission type | UINT8 | RW | FF h |
| 1801 h _03 h | inhibit time (100 μs) | UINT16 | RW | 64 h |
| 1801 h _05 h | event time (1ms) | UINT16 | RW | 0A h |
| 1A01 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1A01 h _01 h | first mapped object | UINT32 | RW | 60640020 h |
| 1A01 h _02 h | second mapped object | UINT32 | RW | 60610010 h |
| 1A01 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1A01 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

3、T-PDO3

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1802 h _00 h | number of entries | UINT8 | RO | 04 h |
| 1802 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000281 h |
| 1802 h _02 h | transmission type | UINT8 | RW | FF h |
| 1802 h _03 h | inhibit time (100 μs) | UINT16 | RW | 64 h |
| 1802 h _05 h | event time (1ms) | UINT16 | RW | 0A h |
| 1A02 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1A02 h _01 h | first mapped object | UINT32 | RW | 60640020 h |
| 1A02 h _02 h | second mapped object | UINT32 | RW | 60610010 h |
| 1A02 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1A02 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

4、T-PDO4

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1803 h _00 h | number of entries | UINT8 | RO | 04 h |
| 1803 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000281 h |
| 1803 h _02 h | transmission type | UINT8 | RW | FF h |
| 1803 h _03 h | inhibit time (100 μs) | UINT16 | RW | 64 h |
| 1803 h _05 h | event time (1ms) | UINT16 | RW | 0A h |
| 1A03 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1A03 h _01 h | first mapped object | UINT32 | RW | 60640020 h |
| 1A03 h _02 h | second mapped object | UINT32 | RW | 60610010 h |
| 1A03 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1A03 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

If transmit tye is 254 (if PDO content has changed,trigger will be sent by PDO),using the following object can shield parts of PDO changers.Only when the un-shield bit has changed，PDO is occur.If wants shielding any bit, the corresponding bit of object write to 0.

tpdo_1_transmit_mask

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 2000 h _00 h | number of entries | UINT8 | RO | 02 h |
| 2000 h _01 h | tpdo_1_transmit_mask_low | UINT32 | RW | FFFFFFFF h |
| 2000 h _02 h | tpdo_1_transmit_mask_high | UINT32 | RW | FFFFFFFF h |

tpdo_2_transmit_mask

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 2001 h _00 h | number of entries | UINT8 | RO | 02 h |
| 2001 h _01 h | tpdo_2_transmit_mask_low | UINT32 | RW | FFFFFFFF h |
| 2001 h _02 h | tpdo_2_transmit_mask_high | UINT32 | RW | FFFFFFFF h |

tpdo_3_transmit_mask

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 2002 h _00 h | number of entries | UINT8 | RO | 02 h |
| 2002 h _01 h | tpdo_1_transmit_mask_low | UINT32 | RW | FFFFFFFF h |
| 2002 h _02 h | tpdo_1_transmit_mask_high | UINT32 | RW | FFFFFFFF h |

tpdo_4_transmit_mask

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 2003 h _00 h | number of entries | UINT8 | RO | 02 h |
| 2003 h _01 h | tpdo_2_transmit_mask_low | UINT32 | RW | FFFFFFFF h |
| 2003 h _02 h | tpdo_2_transmit_mask_high | UINT32 | RW | FFFFFFFF h |

1、R-PDO1

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1400 h _00 h | number of entries | UINT8 | RO | 02 h |
| 1400 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000201 h |
| 1400 h _02 h | transmission type | UINT8 | RW | FF h |
| 1600 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1600 h _01 h | first mapped object | UINT32 | RW | 60400010 h |
| 1600 h _02 h | second mapped object | UINT32 | RW | 60FF0020 h |
| 1600 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1600 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

2、R-PDO2

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1401 h _00 h | number of entries | UINT8 | RO | 02 h |
| 1401 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000301 h |
| 1401 h _02 h | transmission type | UINT8 | RW | FF h |
| 1601 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1601 h _01 h | first mapped object | UINT32 | RW | 60FF0020 h |
| 1601 h _02 h | second mapped object | UINT32 | RW | 60600010 h |
| 1601 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1601 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

3、R-PDO3

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1402 h _00 h | number of entries | UINT8 | RO | 02 h |
| 1402 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000301 h |
| 1402 h _02 h | transmission type | UINT8 | RW | FF h |
| 1602 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1602 h _01 h | first mapped object | UINT32 | RW | 60FF0020 h |
| 1602 h _02 h | second mapped object | UINT32 | RW | 60600010 h |
| 1602 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1602 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

4、R-PDO4

| Index | Comment | Type | Acc. | Default Value |
|---|---|---|---|---|
| 1403 h _00 h | number of entries | UINT8 | RO | 02 h |
| 1403 h _01 h | COB-ID used by PDO | UINT32 | RW | 00000301 h |
| 1403 h _02 h | transmission type | UINT8 | RW | FF h |
| 1603 h _00 h | number of mapped objects | UINT8 | RW | 02 h |
| 1603 h _01 h | first mapped object | UINT32 | RW | 60FF0020 h |
| 1603 h _02 h | second mapped object | UINT32 | RW | 60600010 h |
| 1603 h _03 h | third mapped object | UINT32 | RW | 00 h |
| 1603 h _04 h | fourth mapped object | UINT32 | RW | 00 h |

# 3.4 SYNC message

Synchronization object is used for controlling data synchronize transmit. For example: starting synchronously several axises. The transmition of synchronous message is based on Producer-Customer model. All the nodes of synchronous PDO can receive (at the same time) the message as customer and synchronize other node.

General mode:

CANopen suggests a COB-ID with highest priority to ensure that synchronized signal could be transmitted properly. Without transferring data, SYNC message could be as short as possible.

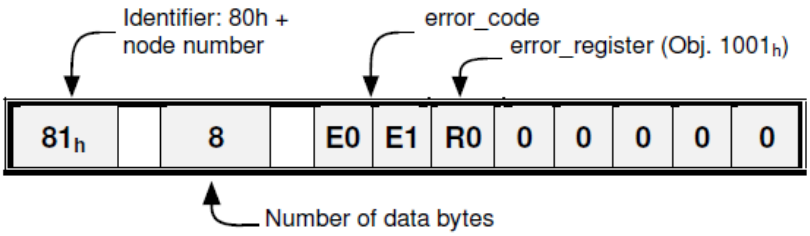
The identifier the servo controller receives SYNC messages are fixed to 080h. The identifier can be read via the object **cob_id_sync**.

| | |
|---|---|
| Index | 1005 $_{h}$ |
| Name | cob_id_sync |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | NO |
| Units | —— |
| Value Range | 80000080 $_{h}$, 00000080 $_{h}$ |
| Default Value | 00000080 $_{h}$ |

# 3.5 Emergency message

When an alarm occurs to drive, CANopen will initiate an Emergency message to inform the current drive type and error code to clients. Error code displayed on panel can be read on low byte of 603Fh object.

The structure of Emergency message：




Alarm code

| error_code（hex） | Description |
|---|---|
| 2310 | Over current |
| 3100 | Instantaneous power failure |
| 3110 | Over voltage |
| 3120 | Under voltage |
| 5080 | RAM exception |
| 5210 | AD sampling error |
| 5420 | Regenerative resistor error |
| 5421 | Regenerative resistor exception |
| 5581 | Parameter checksum exception |
| 5582 | electric gear error |
| 5583 | Motor type or drive type error |
| 6100 | Illegal error code |
| 6120 | PDO mapping error |
| 6300 | CAN communication error(Address or communication baud rate error) |
| 7303 | serial encoder error |
| 7305 | Incremental encoder error |
| 7380 | Resolver error |
| 8100 | CAN communication exception |
| 8110 | CAN bus overflow |
| 8120 | PASSIVE CAN bus turn to PASSIVE |
| 8130 | Heartbeat error |
| 8140 | CAN BUS OFF |
| 8200 | Length of CAN messages error |
| 8210 | Length of receiving PDO error |
| 8311 | Overload alarm |
| 8480 | Over speed alarm |
| 8681 | Forward run prohibited POT |
| 8682 | Reverse run prohibited NOT |

Relevant parameter：

| | |
|---|---|
| Index | 1003 h |
| Name | pre_defined_error_field |
| Object Code | ARRAY |
| No. of Elements | 4 |
| Data Type | UINT32 |

| | |
|---|---|
| Sub-Index | 01 h |
| Description | standard_error_field_0 |
| Access | RO |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | —— |

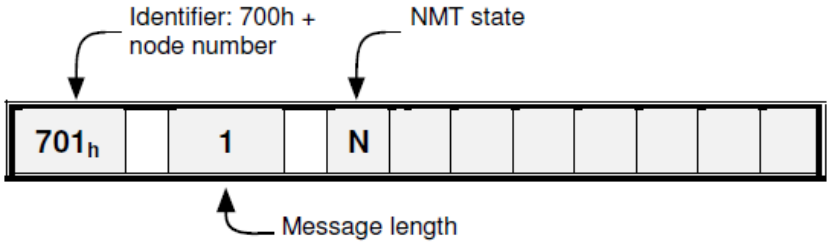
| | |
|---|---|
| Sub-Index | 02 h |
| Description | standard_error_field_1 |
| Access | RO |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | —— |

| | |
|---|---|
| Sub-Index | 03 h |
| Description | standard_error_field_2 |
| Access | RO |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | —— |

| | |
|---|---|
| Sub-Index | 04 h |
| Description | standard_error_field_3 |
| Access | RO |
| PDO Mapping | NO |
| Units | —— |
| Value Range | —— |
| Default Value | —— |

# 3.6 HEARTBEAT message

Structure of the heartbeat message：

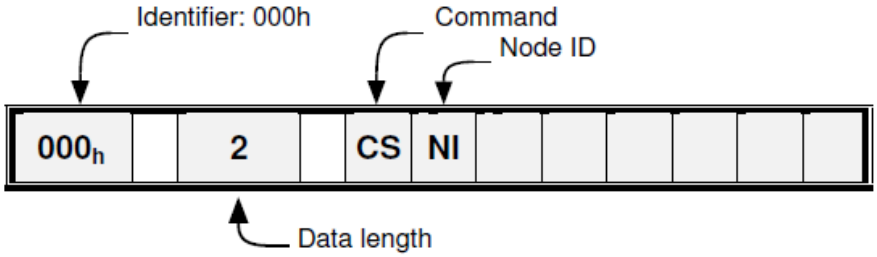



Relevant parameter：

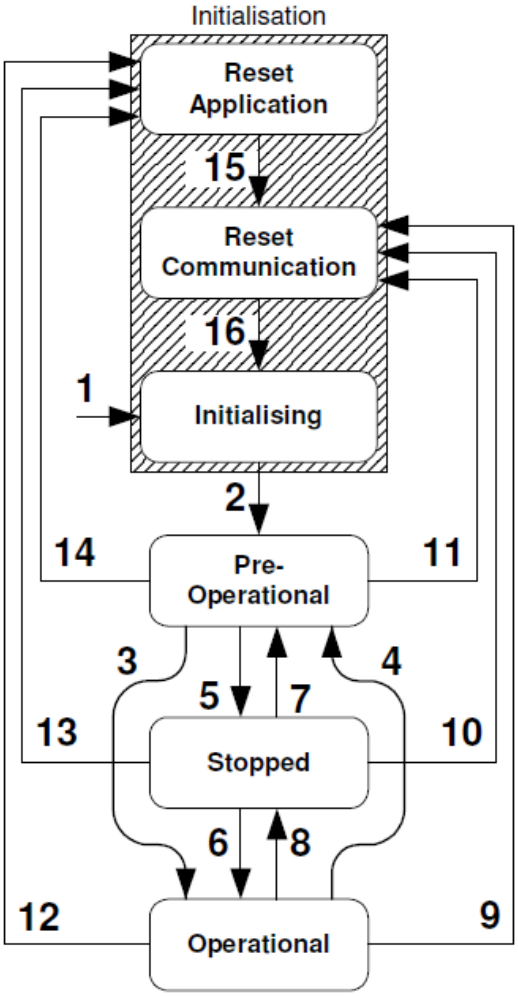
| | |
|---|---|
| Index | 1017 h |
| Name | producer_heartbeat_time |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | ms |
| Value Range | 0 - 65535 |
| Default Value | 0 |

# 3.7 Network management (NMT service)

Structure of the message：




NMT-State machine：

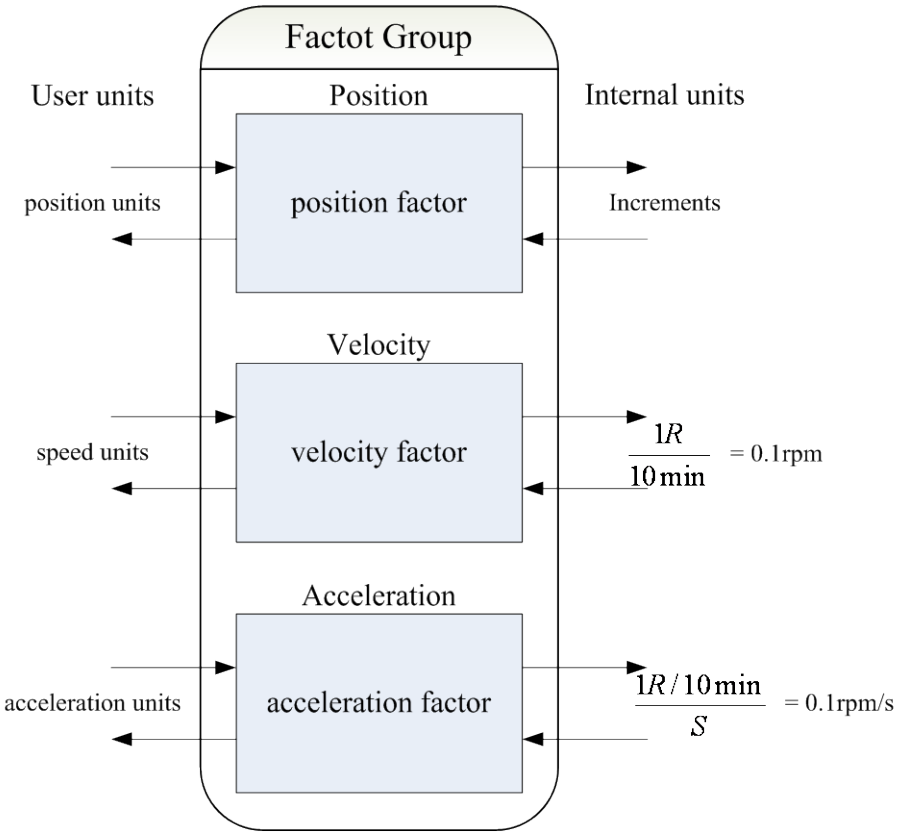


| CS | Meaning | Transition | Target state |
|---|---|---|---|
| $01_h$ | Start Remote Node | 3, 6 | **Operational** |
| $02_h$ | Stop Remote Node | 5, 8 | **Stopped** |
| $80_h$ | Enter Pre-Operational | 4, 7 | **Pre-Operational** |
| $81_h$ | Reset Application | 12, 13, 14 | **Reset Application** |
| $82_h$ | Reset Communication | 9, 10, 11 | **Reset Communication** |

| Name | Meaning | SDO | PDO | NMT |
|---|---|---|---|---|
| **Reset Application** | No communication. All CAN objects are set to their reset values (application parameter set). | - | - | - |
| **Reset Communication** | No communication.<br>The CAN controller will be re-initialised. | - | - | - |
| **Initialising** | State after Hardware Reset. Reset of the CAN node, sending of the Bootup message | - | - | - |
| **Pre-Operational** | Communication via SDOs possible.<br>PDOs inactive (No sending / receiving) | X | - | X |
| **Operational** | Communication via SDOs possible.<br>PDOs active (sending / receiving) | X | X | X |
| **Stopped** | No communication except heartbeat + NMT | - | - | X |

# Chapter 4 Conversion factors(factor group)

Servo controllers will be used in a huge number of applications: As direct drive, with gear or for linear drives. To allow an easy parameterization for all kinds of applications, the servo controller can be parameterized in such a way that all values like the demand velocity refer to the driven side of the plant. The necessary calculation is done by the servo controller.




The default setting of the Factor Group is as follows:

| Value | Name | Unit | Remark |
|---|---|---|---|
| Length | position units | Increments | Increments per revolution * |
| Velocity | speed units | 1R /10min | 0.1rpm |
| Acceleration | Acceleration units | 1R/10min/s | 0.1rpm/s |
| Jerk | jerk units | pulse/(s*100μs*100μs) | Range:1-20,more smaller,more smooth |

*** :** Common incremental encoder: 10000P/R
Resolver: 65536P/R
17 bit incremental encoder: 131072P/R
17 bit absolute encoder: 131072P/R

## 4.1 Related parameters

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 6093 h | ARRAY | position factor | UINT32 | RW |
| 6094 h | ARRAY | velocity factor | UINT32 | RW |
| 6097 h | ARRAY | acceleration factor | UINT32 | RW |

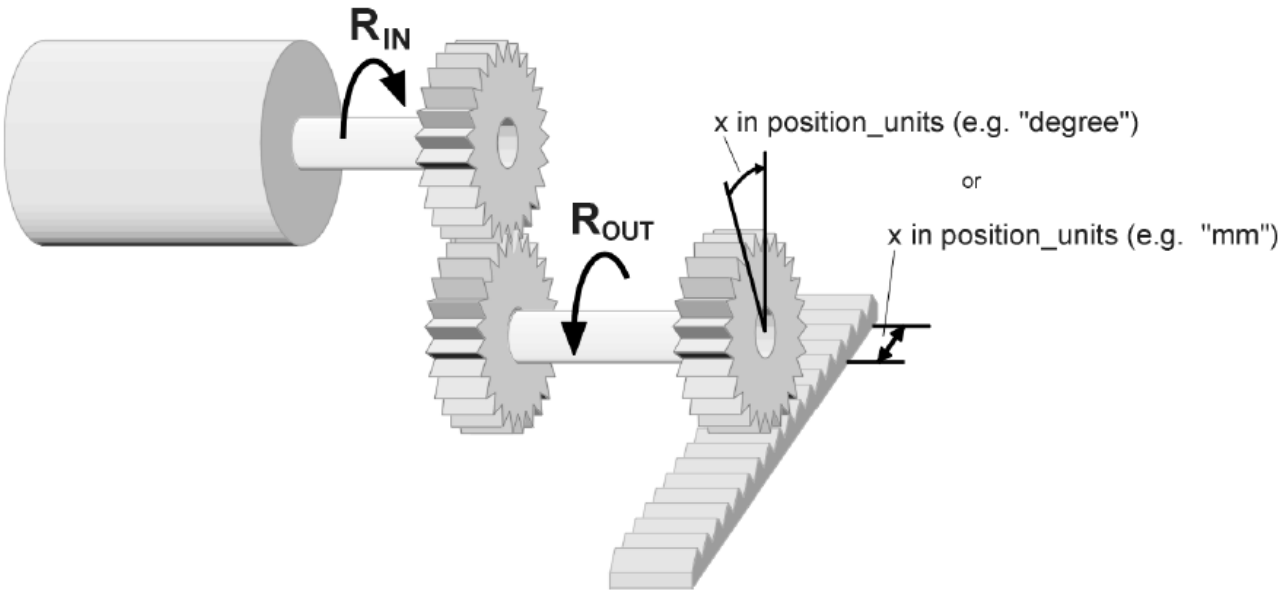
## 4.2 Position factor

The object **position factor** converts all values of length of the application from **Position units** into the internal unit **increments** (*encoder resolution* equals 1 Revolution). It consists of numerator and divisor:

| | |
|---|---|
| Index | 6093 h |
| Name | position factor |
| Object Code | ARRAY |
| No. of Elements | 2 |
| Data Type | UINT32 |

| | |
|---|---|
| Sub-Index | 01 h |
| Description | numerator |
| Access | RW |
| PDO Mapping | YES |
| Units | —— |
| Value Range | —— |
| Default Value | When power on, this value will be initiated to parameter Pn201 |

| | |
|---|---|
| Sub-Index | 02 h |
| Description | division |
| Access | RW |
| PDO Mapping | YES |
| Units | —— |
| Value Range | —— |
| Default Value | When power on, this value will be initiated to parameter Pn202 |




To calculate the **position factor** the following values are necessary:

**gear_ratio** Ratio between revolutions on the driving side (RIN) and revolutions on the driven side (ROUT).

**feed_constant** Ratio between revolutions on the driven side (ROUT) and equivalent motion in **position_units** (e.g. 1 rev = 360°)

The calculation of the **position_factor** is done with the following equation:

$$\textbf{position factor} = \frac{\text{numerator}}{\text{division}} = \frac{\text{gear\_ratio} * \text{encoder\_resolution}}{\text{feed\_constant}}$$

Note:

| Encoder type | Encoder_resolution(Unit: Inc) |
|---|---|
| Common incremental encoder | 10000 |
| Resolver | 65535 |
| 17 bit incremental encoder | 131072 |
| 17 bit absolute encoder: 131072P/R | 131072 |

# 4.3 Velocity factor

The object **velocity factor** converts all speed values of the application from **speed_units** into the internal unit **revolutions 0.1rpm**. It consists of numerator and divisor

| | |
|---|---|
| Index | 6094 $_{h}$ |
| Name | velocity factor |
| Object Code | ARRAY |
| No. of Elements | 2 |
| Data Type | UINT32 |

| | |
|---|---|
| Sub-Index | 01 $_{h}$ |
| Description | numerator |
| Access | RW |
| PDO Mapping | YES |
| Units | —— |
| Value Range | —— |
| Default Value | 1 |

| | |
|---|---|
| Sub-Index | 02 $_{h}$ |
| Description | division |
| Access | RW |
| PDO Mapping | YES |
| Units | —— |
| Value Range | —— |
| Default Value | 1 |

In principle the calculation of the **velocity factor** is composed of two parts: A conversion factor from internal units of length into **position_units** and a conversion factor from internal time units into user defined time units (e.g. from seconds to minutes). The first part equals the calculation of the **position_factor.** For the second part another factor is necessary for the calculation:

**time_factor_v** Ratio between internal and user defined time units.
(z.B. **1** ***min*** = **1/10** ***10 min***)

**gear_ratio** Ratio between revolutions on the driving side (RIN) and revolutions on the driven side (ROUT).

**feed_constant** Ratio between revolutions on the driven side (ROUT) and equivalent motion in position_units (e.g. 1 R = 360°)

The calculation of the **velocity factor** is done with the following equation:

$$\textbf{velocity factor} = \frac{\text{numerator}}{\text{division}} = \frac{\text{gear\_ratio} * \text{time\_factor\_v}}{\text{feed\_constant}}$$

# 4.4 Acceleration factor

The object **acceleration_factor** converts all acceleration values of the application from **acceleration_units** into the internal unit（0.1rpm）。It consists of numerator and divisor:

| | |
|---|---|
| Index | 6094 $_{h}$ |
| Name | acceleration factor |
| Object Code | ARRAY |
| No. of Elements | 2 |
| Data Type | UINT32 |

| | |
|---|---|
| Sub-Index | 01 $_{h}$ |
| Description | numerator |
| Access | RW |
| PDO Mapping | YES |
| Units | —— |
| Value Range | —— |
| Default Value | 1 |

| | |
|---|---|
| Sub-Index | 02 $_{h}$ |
| Description | division |
| Access | RW |
| PDO Mapping | YES |
| Units | —— |
| Value Range | —— |
| Default Value | 1 |

The calculation of the **acceleration_factor** is also composed of two parts: A conversion factor from internal units of length into **position_units** and a conversion factor from internal time units squared into user defined time units squared (e.g. from seconds2 to minutes2). The first part equals the calculation of the **position_factor.** For the second part another factor is necessary for the calculation

**time_factor_a** Ratio between internal time units squared and user defined time units squared

（z.B.： 1**min**$^2$ = 1min*min = 60s*1min =60/10 **10min/s**）

**gear_ratio** Ratio between revolutions on the driving side (RIN) and revolutions on the driven side (ROUT).

**feed_constant** Ratio between revolutions on the driven side (ROUT) and equivalent motion in position_units (e.g. 1 R = 360°)

The calculation of the **acceleration_factor** is done with the following equation:

$$\textbf{acceleration factor} = \frac{\text{numerator}}{\text{division}} = \frac{\text{gear\_ratio} * \text{time\_factor\_a}}{\text{feed\_constant}}$$

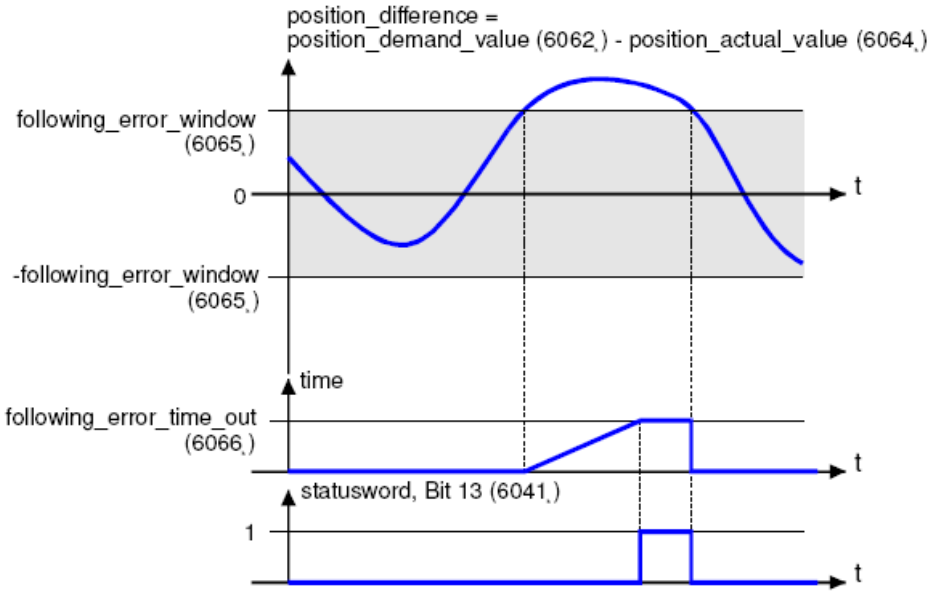
# Chapter 5 Position control function

This chapter describes all parameters which are required for the position controller. The desired position value (**position_demand_value**) of the trajectory generator is the input of the position controller. Besides this the actual position value (**position_actual_value**) is supplied by the angle encoder (resolver, incremental encoder, etc.). The behaviour of the position controller can be influenced by parameters.

It is possible to limit the output quantity (**control_effort**) in order to keep the position control system stable. The output quantity is supplied to the speed controller as desired speed value. In the **Factor Group** all input and output quantities are converted from the application-specific units to the respective internal units of the controller

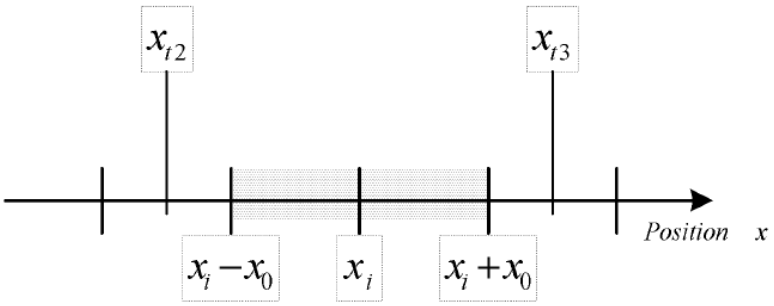
The following subfunctions are defined in this chapter:

1. Trailing error (Following Error)




Trailing error (Following Error) – Function Survey

The deviation of the actual position value (**position_actual_value**) from the desired position value (**position_demand_value)** is named trailing error. If for a certain period of time this trailing error is bigger than specified in the trailing error window (**following_error_window**) bit 13 (**following_error**) of the object **statusword** will be set to 1.



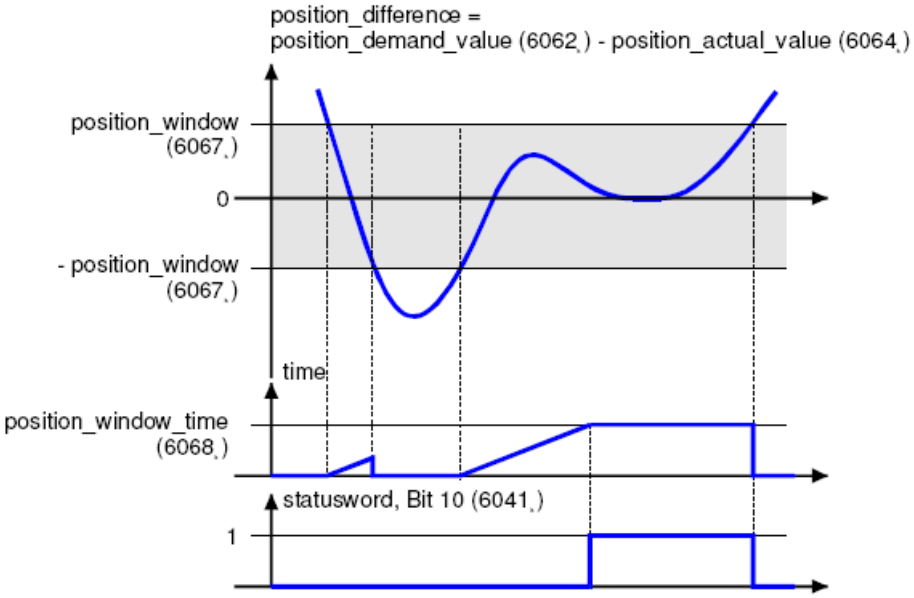

Trailing error (following error)

The permissible time can be defined via the object **following_error_time_out**.。 Figure above shows how the window function is defined for the message "following error".The range between xi-x0 and xi+x0 is defined symmetrically

around the desired position (**position_demand_value**) xi. For example the positions xt2 and xt3 are outside this window (**following_error_window**). If the drive leaves this window and does not return to the window within the time defined in the object **following_error_time_out** then bit 13 (**following_error**) in the **statusword** will be set to 1.

2**.** Position Reached

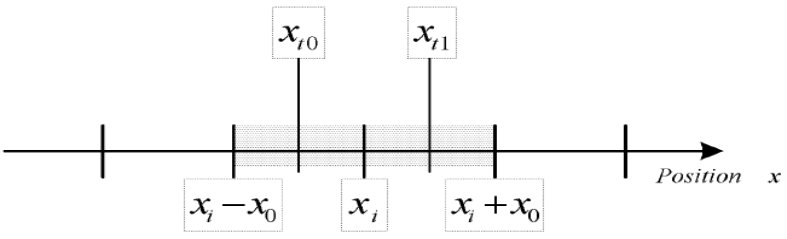
This function offers the chance to define a position window around the target position (**target_position**). If the actual position of the drive is within this range for a certain period of time – the **position_window_time** – bit 10 (**target_reached**) will be set to 1 in the statusword.




.

Position reached-function description

Figure below shows how the window function is defined for the message ”position reached”. The position range between xi-x0 and xi+x0 is defined symmetrically around the target position (**target_position**) xi. For example the positions xt0 and xt1 are inside this position window (**position_window**). If the drive is within this window a timer is started. If this timer reaches the time defined in the object **position_window_time** and the drive uninterruptedly was within the valid range between xi-x0 and xi+x0, bit 10 (**target_reached**) will be set in the **statusword**. As far as the drive leaves the permissible range, bit 10 is cleared and the timer is set to zero.




.

Position reached

Parameters:

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 6062 h | VAR | position_demand_value | INT32 | RO |
| 6063 h | VAR | position_actual_value* | INT32 | RO |
| 6064 h | VAR | position_actual_value | INT32 | RO |
| 6065 h | VAR | following_error_window | UINT32 | RW |
| 6066 h | VAR | following_error_time_out | UINT16 | RW |
| 6067 h | VAR | position_window | UINT32 | RW |
| 6068 h | VAR | position_time | UINT16 | RW |
| 60FA h | VAR | control_effort | INT32 | RO |

| | |
|---|---|
| Index | 6062 h |
| Name | position_demand_value |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RO |
| PDO Mapping | YES |
| Units | position units |
| Value Range | -- |
| Default Value | -- |

| | |
|---|---|
| Index | 6064 h |
| Name | position_ actual _value |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RO |
| PDO Mapping | YES |
| Units | position units |
| Value Range | -- |
| Default Value | -- |

| | |
|---|---|
| Index | 6065 h |
| Name | following_error_window |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | position units |
| Value Range | 0 – 7FFFFFFF h |
| Default Value | 256 |

| Index | 6066 h |
|---|---|
| Name | following_error_time_out |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | Ms |
| Value Range | 0 – 65535 |
| Default Value | 0 |

| Index | 60FA h |
|---|---|
| Name | control_effort |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RO |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | -- |

| Index | 6067 h |
|---|---|
| Name | position_window |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | position units |
| Value Range | -- |
| Default Value | 400 |

| Index | 6068 h |
|---|---|
| Name | position_time |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | Ms |
| Value Range | 0 – 65535 |
| Default Value | 0 |

# Chapter 6 Device control

The following chapter describes how to control the servo controller using CANopen, i.e. how to switch on the power stage or to reset an error.

## 6.1 State diagram (State machine)

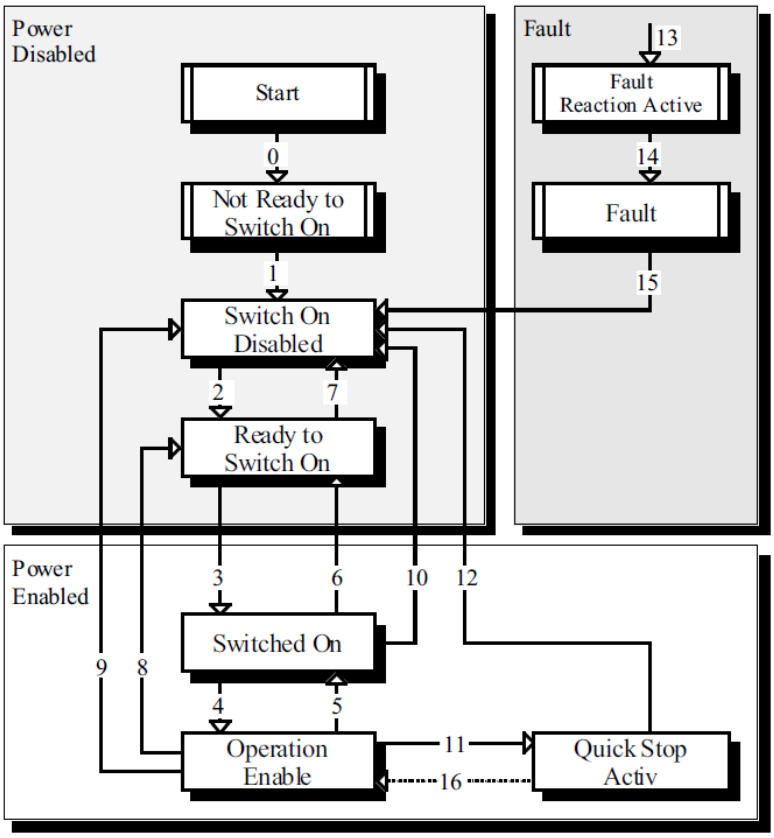
Using CANopen the complete control of the servo is done by two objects. Via the **controlword** the host is able to control the servo, as the status of the servo can be read out of the **statusword**. The following items will be used in this chapter:

State： The servo controller is in different states dependent on for instance if the power stage is alive or if an error has occurred. States defined under CANopen will be explained in this chapter.
Example: **SWITCH_ON_DISABLED**

State Transition： Just as the states it is defined as well how to move from one state to another (e.g. to reset an error). These state transitions will be either executed by the host by setting bits in the **controlword** or by the servo controller itself, if an error occurs for instance.

Command： To initiate a state transition defined bit combinations have to be set in the **controlword**. Such bit combination are called command. Example: **Enable Operation**

State diagram： All the states and all state transitions together form the so called state diagram: A survey of all states and the possible transitions between two states.。



State diagram of the servo controller

The state diagram can be divided into three main parts: "Power Disabled" means the power stage is switched off and "Power Enabled" the power stage is active. The area "Fault" contains all states necessary to handle errors of the controller. The most important states have been highlighted in the Figure: After switching on the servo controller initializes itself and reaches the state **SWITCH_ON_DISABLED** after all. In this state CAN communication is possible and the servo controller can be parameterized (e.g. the mode of operation can be set to "velocity control"). The power stage remains switched off and the motor shaft is freely rotatable. Through the state transitions 2, 3 and 4 – principally like the controller enable under CANopen - the state **OPERATION_ENABLE** will be reached. In this state the power stage is live and the servo controller controls the motor according to the parameterized mode of operation. Therefore previously ensure that the servo controller has been parameterized correctly and the according demand value is zero. The state transition 9 complies with disabling the power stage, i.e. the motor is freely rotatable.

| Status | Description |
|---|---|
| **Not Ready to Switch On** | The servo controller executes its self-test. The CAN communication is not working |
| **Switch On Disabled** | The self-test has been completed. The CAN communication is activated.. |
| **Ready to Switch On** | Servo driver is waiting for the state of Switch and servo motor is not at power stage |
| **Switched On** | The power stage is alive. |
| **Operation Enable** | The motor is under voltage and is controlled according to operational mode |
| **Quick Stop Active** | Servo driver will be stopped through its fixed way, |
| **Fault Reaction Active** | Servo driver tests error and will be stopped through its fixed way, with motor's power stage alive |
| **Fault** | An error has occurred. The power stage has been switched off. |

# 6.2 Relevant parameters of device control

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 6040 $_h$ | VAR | controlword | UINT16 | RW |
| 6041 $_h$ | VAR | statusword | UINT16 | RO |
| 605A $_h$ | VAR | quick_stop_option_code | INT16 | RW |
| 605B $_h$ | VAR | shutdown_option_code | INT16 | RW |
| 605C $_h$ | VAR | disabled_operation_option_code | INT16 | RW |
| 605D $_h$ | VAR | halt_option_code | INT16 | RW |
| 605E $_h$ | VAR | fault_reaction_option_code | INT16 | RW |

## 6.2.1 Controlword

| | |
|---|---|
| Index | 6040 $_h$ |
| Name | controlword |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |

| | |
|---|---|
| Units | -- |
| Value Range | -- |
| Default Value | 0 |

Controlword bit description is as below…

| 15 11 | 10 9 | 8 | 7 | 6 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| manufacturer specific | reserved | halt | Fault reset | Operation mode specific | Enable operation | Quick stop | Enable voltage | Switch on |

Bit0 ~ 3 and Bit7：

Transmit of status machine is triggered by 5 bits coordinated control code as below…

| Command | Bit of the *controlword* | | | | | Transitions |
|---|---|---|---|---|---|---|
| | Fault reset | Enable operation | Quick stop | Enable voltage | Switch on | |
| Shutdown | 0 | X | 1 | 1 | 0 | 2,6,8 |
| Switch on | 0 | 0 | 1 | 1 | 1 | 3* |
| Switch on | 0 | 1 | 1 | 1 | 1 | 3** |
| Disable voltage | 0 | X | X | 0 | X | 7,9,10,12 |
| Quick stop | 0 | X | 0 | 1 | X | 7,10,11 |
| Disable operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable operation | 0 | 1 | 1 | 1 | 1 | 4,16 |
| Fault reset |  | X | X | X | X | 15 |

Device control list

Note: X means this bit could be ignored.

Bit4、5、6、8：

The definition of this 4 bit is different in different control mode…

| Bit | Control Mode | | |
|---|---|---|---|
| | profile position mode | profile velocity mode | homing mode |
| 4 | new_set_point | reserved | start_homeing_operation |
| 5 | change_set_immediatly | reserved | reserved |
| 6 | abs/rel | reserved | reserved |
| 8 | Halt | Halt | Halt |

Other bits: all reserved

## 6.2.2 Statusword

| | |
|---|---|
| Index | 6041 $_{h}$ |
| Name | statusword |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RO |
| PDO Mapping | YES |

| | |
|---|---|
| Units | -- |
| Value Range | -- |
| Default Value | -- |

Explanation of statusword bit is as below:

| bit | name |
|---|---|
| 0 | Ready to switch on |
| 1 | Switched on |
| 2 | Operation enabled |
| 3 | Fault |
| 4 | Voltage enabled |
| 5 | Quick stop |
| 6 | Switch on disabled |
| 7 | Warning |
| 9~8 | Not used now |
| 10 | Target reached |
| 11 | Internal limit active |
| 13~12 | Operation mode specific |
| 15~14 | Not used now |

Bit0 ~ 3 , Bit5 and Bit6：

The combination of this bit indicates the status of drives.

| Value (binary) | State |
|---|---|
| xxxx xxxx x0xx 0000 | Not ready to switch on |
| xxxx xxxx x1xx 0000 | Switch on disabled |
| xxxx xxxx x01x 0001 | Ready to switch on |
| xxxx xxxx x01x 0011 | Switched on |
| xxxx xxxx x01x 0111 | Operation enabled |
| xxxx xxxx x00x 0111 | Quick stop active |
| xxxx xxxx x0xx 1111 | Fault reaction active |
| xxxx xxxx x0xx 1000 | Fault |

Bit4： Voltage enabled

Main power is on when this bit is 1.

Bit5： Quick stop

Driver will follow setting （605A $_h$：quick_stop_option_code）to halt when this bit is 0.

Bit7： Warning

Driver detects alarm when this bit is 1.

Bit10： Target reached

In different control modes the meaning of this bit is different.

In profile position mode, when set position is reached, this bit is set. When Halt is booted, speed is reduced to 0 and this bit will be set. When new position is set, this bit will be cleared.

In profile Velocity Mode, when the speed reaches the targeted speed, this bit will be set. When Halt is booted and speed is reduced to 0, this bit is set.

Bit11： Internal limit active

When this bit is 1, it indicates that internal torque has surpassed the set value,or reached the max.forward/reverse run.It can be confirmed by reading object 60FDh（digital inputs）.

Bit12, 13：

These 2 bits mean different in different control mode…

| Bit | Control mode | | |
|---|---|---|---|
| | profile position mode | profile velocity mode | homing mode |
| 12 | Set-point acknowledge | Speed | Homing attained |
| 13 | Following error | Max slippage error | Homing error |

Other bits：

All reserved.

## 6.2.3 Shutdown_option_code

The object **shutdown_option_code** determines the behaviour if the state transition 8 (from **OPERATION ENABLE** to **READY TO SWITCH ON)** will be executed.

| | |
|---|---|
| Index | 605B $_{h}$ |
| Name | shutdown_option_code |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | -- |
| Value Range | 0,1 |
| Default Value | 0 |

| Value | Name |
|---|---|
| 0 | Power stage will be switched off. Motor is freely rotatable. |
| 1 | Switch off the power stage after the motor stops deceleration. |

## 6.2.4 Disable_operation_option_code

The object **disable_operation_option_code** determines the behaviour if the state transition 5 (from **OPERATION ENABLE** to **SWITCHED ON)** will be executed.

| Index | 605C $_{h}$ |
|---|---|
| Name | disable_operation_option_code |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | -- |
| Value Range | 0,1 |
| Default Value | 0 |

| Value | Description |
|---|---|
| 0 | Power stage will be switched off. Motor is freely rotatable.。 |
| 1 | Switch off the power stage after the motor stops deceleration. |

## 6.2.5 Quick_stop_option_code

The object **quick_stop_option_code** determines the behaviour if a **Quick Stop** will be executed.

| Index | 605A $_{h}$ |
|---|---|
| Name | quick_stop_option_code |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | -- |
| Value Range | 0,1,2,5,6 |
| Default Value | 0 |

| value | Description |
|---|---|
| 0 | Power stage will be switched off. Motor is freely rotatable. |
| 1 | Switch off the power stage after the motor stops deceleration. |
| 2 | Power stage will be shut down after the motor decelerates to still urgently. |
| 5 | QuickStop is alive after the motor decelerates to still. |
| 6 | QuickStop is alive after the motor decelerates urgently to still. |

## 6.2.6 Halt_option_code

Halt_option_code determines how to stop when bit.8 (halt) of controlword is set to 1.

| | |
|---|---|
| Index | 605D $_{h}$ |
| Name | halt_option_code |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | -- |
| Value Range | 1,2 |
| Default Value | 0 |

| Value | Description |
|---|---|
| 1 | The motor decelerates to still. |
| 2 | The motor decelerates urgently to still |

## 6.2.7 Fault_reaction_option_code

When an error is occurred, fault_reation_option_code determines how to stop.

| | |
|---|---|
| Index | 605D $_{h}$ |
| Name | fault_reaction_option_code |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | NO |
| Units | -- |
| Value Range | 0 |
| Default Value | 0 |

| Value | Description |
|---|---|
| 0 | Power stage will be switched off. Motor is freely rotatable. |

# Chapter 7 Control mode

PRONET drive currently supports 3 control modes in CANopen DSP402：

HOMING MODE

PROFILE VELOCITY MODE

PROFILE POSITION MODE

This chapter mainly describes three control modes as above.

## 7.1 Relevant parameter of control mode

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 6060 h | VAR | modes_of_operation | INT8 | RW |
| 6061 h | VAR | modes_of_operation_display | INT8 | RO |

### 7.1.1 Modes_of_operation

Drive control mode will be determined by parameters in modes_of_operation.

| | |
|---|---|
| Index | 6060 h |
| Name | modes_of_operation |
| Object Code | VAR |
| Data Type | INT8 |
| Access | RW |
| PDO Mapping | YES |
| Units | -- |
| Value Range | 1,3,6 |
| Default Value | 0 |

| Value | Description |
|---|---|
| 0 | NOP MODE |
| 1 | PROFILE POSITION MODE |
| 3 | PROFILE VELOCITY MODE |
| 6 | HOMING MODE |

### 7.1.2 Modes_of_operation_display

Drive current control mode could be read from parameters in modes_of_operation_display.

| | |
|---|---|
| Index | 6061 h |
| Name | modes_of_operation_display |
| Object Code | VAR |
| Data Type | INT8 |
| Access | RO |
| PDO Mapping | YES |
| Units | -- |
| Value Range | 1,3,6 |
| Default Value | 0 |

Note：The current control mode could be only known from parameters in modes_of_operation_display

## 7.2 Homing mode

PRONET servo drive currently supports multiple homing mode and users could choose the suitable homing mode. For example, if an incremental encoder is applied in servomotor, then homing mode of Zero impulse could be chosen and if serial encoder or resolver is applied in servomotor then Zero impulse homing mode couldn't be selected.

The user can determine the velocity, acceleration, and the kind of homing operation. After the servo controller has found its reference the zero position can be moved to the desired point via the object home_offset （607C h）.

### 7.2.1 Control word of homing mode

| 15 ~ 9 | 8 | 7 ~ 5 | 4 | 3 ~ 0 |
|---|---|---|---|---|
| * | Halt | * | home_start_operation | * |

*: referred to previous chapters …

| Name | Value | Description |
|---|---|---|
| Homing operation start | 0 | Homing mode inactive |
| | 0 → 1 | Start homing mode |
| | 1 | Homing mode active |
| | 1 → 0 | Interrupt homing mode |
| Halt | 0 | Execute the instruction of bit 4 |
| | 1 | Stop axle with homing acceleration |

### 7.2.2 Status word of homing mode

| 15 ~ 14 | 13 | 12 | 11 | 10 | 9 ~ 0 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| * | homing_error | homing_attained | * | target_reached | * |

*: referred to previous chapters

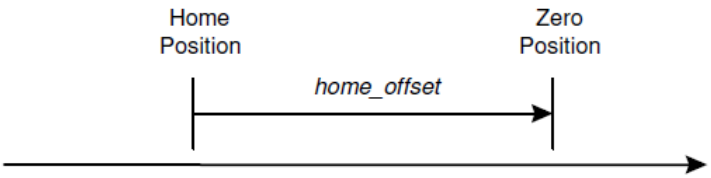
| Name | Value | Description |
|---|---|---|
| Target reached | 0 | Halt = 0: Home position not reached<br>Halt = 1: Axle decelerates |
| | 1 | Halt = 0: Home position reached<br>Halt = 1: Axle has velocity 0 |
| Homing attained | 0 | Homing mode not yet completed |
| | 1 | Homing mode carried out successfully |
| Homing error | 0 | No homing error |
| | 1 | Homing error occurred;<br>Homing mode carried out not successfully;<br>The error cause is found by reading the error code |

## 7.2.3 Relevant parameter of homing mode

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 607C h | VAR | home_offset | INT32 | RW |
| 6098 h | VAR | homing_method | INT8 | RW |
| 6099 h | ARRAY | homing_speeds | UINT32 | RW |
| 609A h | VAR | homing_acceleration | INT32 | RW |

**home_offset**

The object **home_offset** determines the displacement of the zero position to the limit resp. reference switch position.




| | |
|---|---|
| Index | 607C h |
| Name | home_offset |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | position units |
| Value Range | -- |
| Default Value | 0 |

**homing_method**

The negative and positive limit switch, the reference switch and the (periodic) zero impulse of the angle encoder.

| | |
|---|---|
| Index | 6098 h |
| Name | homing_method |
| Object Code | VAR |
| Data Type | INT8 |
| Access | RW |
| PDO Mapping | YES |
| Units | -- |
| Value Range | 1,2,3,4,17,18,19,20 |
| Default Value | 1 |

Homing method value description

| Value | Direction | Target | Reference point for Home position | DS402 |
|---|---|---|---|---|
| 1 | Negative | NOT | Zero impulse | 1 |
| 2 | Positive | POT | Zero impulse | 2 |
| 3 | Negative | Reference switch | Zero impulse | 3 |
| 4 | Positive | Reference switch | Zero impulse | 4 |
| 17 | Negative | NOT | NOT | 17 |
| 18 | Positive | POT | POT | 18 |
| 19 | Negative | Reference switch | Reference switch | 19 |
| 20 | Positive | Reference switch | Reference switch | 20 |

**homing_speeds**

There are two kinds of speeds required to find reference point, speed during search for switch and speed during search for zero.

| | |
|---|---|
| Index | 6099 h |
| Name | homing_speeds |
| Object Code | ARRAY |
| No. of Elements | 2 |
| Data Type | INT32 |

| | |
|---|---|
| Sub-Index | 01 h |
| Name | speed_during_search_for_switch |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 0 |

| | |
|---|---|
| Sub-Index | 02 h |
| Name | speed_during_search_for_zero |
| Object Code | VAR |

| | |
|---|---|
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 0 |

**homing_acceleration**

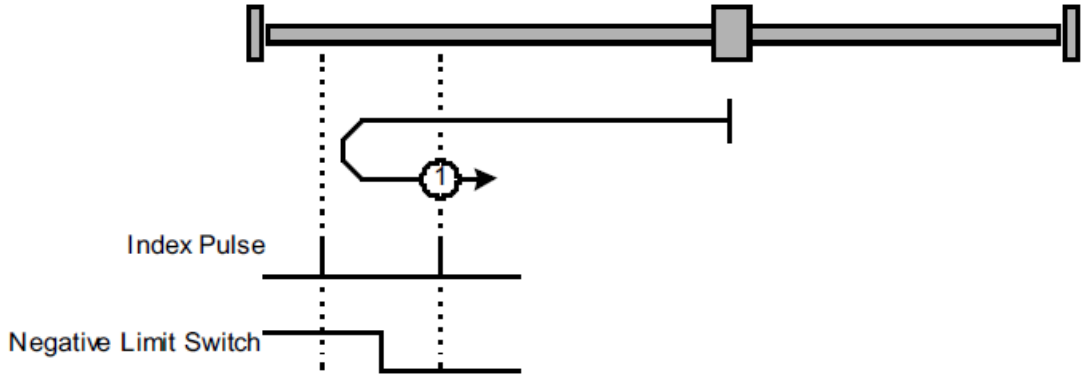
The objects **homing_acceleration** determine the acceleration which is used for all acceleration and deceleration operations during the search for reference.

| | |
|---|---|
| Index | 609A $_{h}$ |
| Name | homing_acceleration |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | acceleration units |
| Value Range | -- |
| Default Value | 0 |

## 7.2.4 Homing sequences

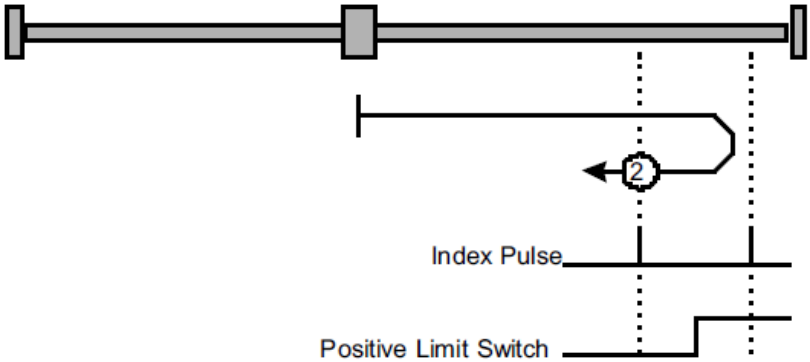
**Method 1: Negative limit switch using zero impulse evaluation**

If this method is used the drive first moves relatively quick into the negative direction until it reaches the negative limit switch. This is displayed in the diagram by the rising edge. Afterwards the drive slowly returns and searches for the exact position of the limit switch. The zero position refers to the first zero impulse of the angle encoder in positive direction from the limit switch.。



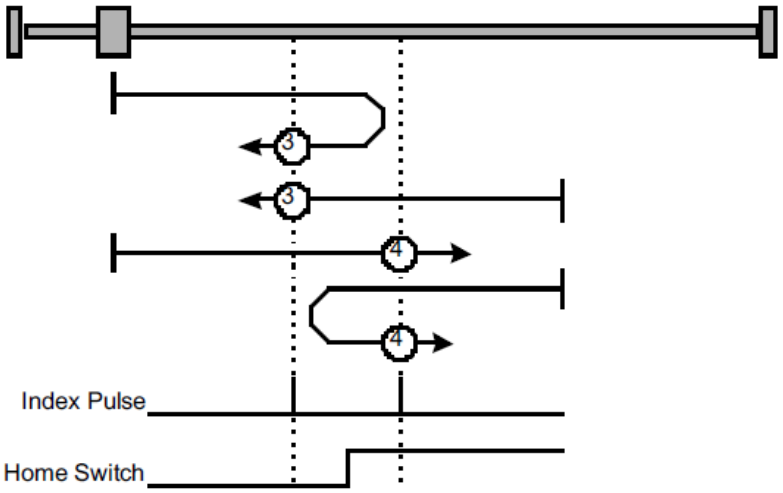

**Method 2: Positive limit switch using zero impulse evaluation**

If this method is used the drive first moves relatively quick into the positive direction until it reaches the positive limit switch. This is displayed in the diagram by the rising edge. Afterwards the drive slowly returns and searches for the exact position of the limit switch. The zero position refers to the first zero impulse of the angle encoder in negative direction from the limit switch.



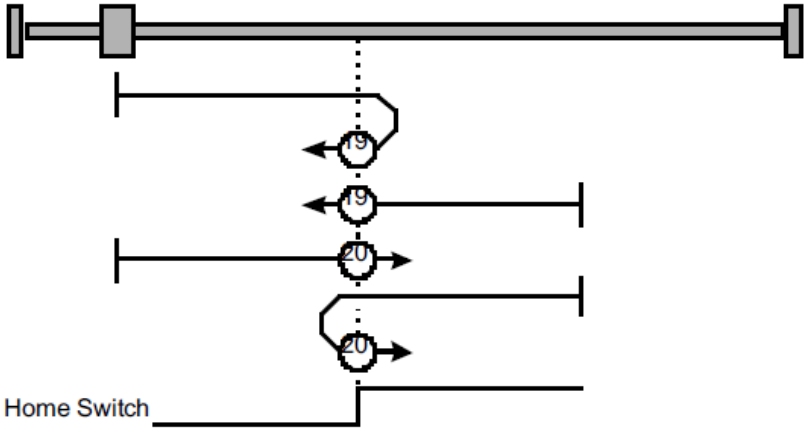

**Methods 3 and 4: Reference switch and zero impulse evaluation**

These two methods use the reference switch which is only active over parts of the distance. These reference methods are particularly useful for round-axis applications where the reference switch is activated once per revolution. In case of method 3 the drive first moves into positive and in case of method 4 into negative direction. Depending on the direction of the motion the zero position refers to the first zero impulse in negative or positive direction from the reference switch. This can be seen in the two following diagrams.

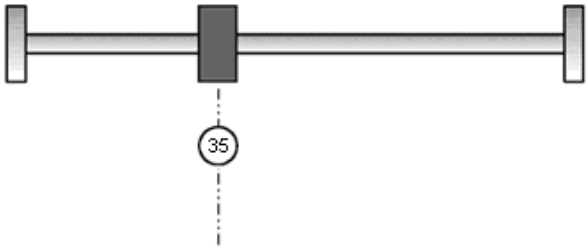



**Method 17~20: Homing operation to the negative limit switch**

If this method is used the drive first moves relatively quick into the negative direction until it reaches the negative limit switch. This is displayed in the diagram by the rising edge. Afterwards the drive slowly returns and searches for the exact position of the limit switch. The zero position refers to the descending edge from the negative limit switch.




**Method 35: set current position as the homing point.**




Notes: When starting homing on homing methed about input signal, the rotation direction of servo motor is associated with the initial status of the input signal.Changing the inicial status by inverse input on set Pn516/Pn517 if it is necessary.When using reference switch homing,I/O should be set as C:HmRef by Pn509/Pn510.Setting parameters refer to <ProNet series AC servo user's manual>.

# 7.3 Profile velocity mode

## 7.3.1 Control word of profile velocity mode

| 15 ~ 9 | 8 | 7 ~ 4 | 3 ~ 0 |
|---|---|---|---|
| * | Halt | * | * |

*: referred to previous chapters

| Name | Value | Description |
|---|---|---|
| Halt | 0 | Execute the motion |
| | 1 | Stop axle |

## 7.3.2 Status word of velocity mode

| 15 ~ 14 | 13 | 12 | 11 | 10 | 9 ~ 0 |
|---|---|---|---|---|---|
| * | MaxSlippageError | Speed | * | Target reached | * |

*: Referred to previous chapters

| Name | Value | Description |
|---|---|---|
| Target reached | 0 | Halt = 0: *Target velocity* not (yet) reached<br>Halt = 1: Axle decelerates |
| | 1 | Halt = 0: *Target velocity* reached<br>Halt = 1: Axle has velocity 0 |
| Speed | 0 | Speed is not equal 0 |
| | 1 | Speed is equal 0 |
| Max slippage error | 0 | Maximum slippage not reached |
| | 1 | Maximum slippage reached |

## 7.3.3 Relevant parameters of profile velocity mode

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 6069 h | VAR | velocity_sensor_actual_value | INT32 | RO |
| 606B h | VAR | velocity_demand_value | INT32 | RO |
| 606C h | VAR | velocity_actual_value | INT32 | RO |
| 609D h | VAR | velocity_window | UINT16 | RW |
| 606E h | VAR | velocity_window_time | UINT16 | RW |
| 606F h | VAR | velocity_threshold | UINT16 | RW |
| 6070 h | VAR | velocity_threshold_time | UINT16 | RW |
| 60FF h | VAR | target_velocity | INT32 | RW |

**velocity_sensor_actual_value**

The speed encoder is read via the object **velocity_sensor_actual_value**. The value is normalised in internal units. The velocity demand value can be read via this object.

| | |
|---|---|
| Index | 6069 h |
| Name | velocity_sensor_actual_value |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | 0.1rmps (1R/10min) |
| Value Range | -- |
| Default Value | -- |

**velocity_demand_value**

The velocity demand value can be read via this object. The unit of this object is the unit of user's speed unit. The velocity demand value can be read via this object.

| | |
|---|---|
| Index | 606B h |
| Name | velocity_demand_value |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RO |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | -- |

**velocity_actual_value**

The actual velocity value can be read via the object **velocity_actual_value**。The velocity demand value can be read via this object.

| | |
|---|---|
| Index | 606C $_{h}$ |
| Name | velocity_actual_value |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RO |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | -- |

**velocity_window**

With the object **velocity_window** a tolerance window for the velocity actual value will be defined for comparing the **velocity_actual_value**（606C $_{h}$）with the target velocity (**target_velocity** object **60FFh**). If the difference is smaller than the velocity window （606D $_{h}$） for a longer time than specified by the object **velocity_window_time** （606E $_{h}$） bit 10 (**target_reached)** will be set in the object **statusword**.

| | |
|---|---|
| Index | 606D $_{h}$ |
| Name | velocity_window |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 20 R/10min |

**velocity_window_time**

The object **velocity_window_time** serves besides the object **606Dh**: **velocity_window** to adjust the window comparator.

| | |
|---|---|
| Index | 606E $_{h}$ |
| Name | velocity_window_time |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | ms |
| Value Range | -- |
| Default Value | 0 |

**velocity_threshold**

The object **velocity_threshold** determines the velocity underneath the axis is regarded as stationary. As soon as the **velocity_actual_value** exceeds the **velocity_threshold** longer than the **velocity_threshold_time** bit 12 is cleared in the **statusword**.

| | |
|---|---|
| Index | 606F $_{h}$ |
| Name | velocity_threshold |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 10 R/10min |

**velocity_threshold_time**

The object **velocity_threshold** determines the velocity below the axis is regarded as stationary. Its unit is ms。As soon as the **velocity_actual_value** exceeds the **velocity_threshold** longer than the **velocity_threshold_time** bit 12 is cleared in the **statusword**.

| | |
|---|---|
| Index | 6070 $_{h}$ |
| Name | velocity_threshold_time |
| Object Code | VAR |
| Data Type | UINT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | ms |
| Value Range | -- |
| Default Value | 0 |

**target_velocity**

The object **target_velocity** is the setpoint for the ramp generator.

| | |
|---|---|
| Index | 60FF $_{h}$ |
| Name | target_velocity |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 0 |

# 7.4 Profile position mode

## 7.4.1 Control word of profile position mode

| 15 ~ 9 | 8 | 7 | 6 | 5 | 4 | 3 ~ 0 |
|---|---|---|---|---|---|---|
| * | Halt | * | abs / rel | Change set immediately | New set-point | * |

*: referred to previous chapter

| Name | Value | Description |
|---|---|---|
| New set-point | 0 | Does not assume *target position* |
| | 1 | Assume *target position* |
| Change set immediately | 0 | Finish the actual positioning and then start the next positioning |
| | 1 | Interrupt the actual positioning and start the next positioning |
| abs / rel | 0 | *Target position* is an absolute value |
| | 1 | *Target position* is a relative value |
| Halt | 0 | Execute positioning |
| | 1 | Stop axle with *profile deceleration* (if not supported with *profile acceleration*) |

## 7.4.2 Status word of profile position mode

| 15 ~ 14 | 13 | 12 | 11 | 10 | 9 ~ 0 |
|---|---|---|---|---|---|
| * | Following error | Set_point acknowledge | * | Target reached | * |

*: referred to previous chapter

| Name | Value | Description |
|---|---|---|
| Target reached | 0 | Halt = 0: *Target position* not reached<br>Halt = 1: Axle decelerates |
| | 1 | Halt = 0: *Target position* reached<br>Halt = 1: Velocity of axle is 0 |
| Set-point acknowledge | 0 | Trajectory generator has not assumed the positioning values (yet) |
| | 1 | Trajectory generator has assumed the positioning values |
| Following error | 0 | No following error |
| | 1 | Following error |

## 7.4.3 Revelant parameters of profile position mode

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 607A $_h$ | VAR | target_position | INT32 | RW |
| 6081 $_h$ | VAR | profile_velocity | UINT32 | RW |
| 6082 $_h$ | VAR | end_velocity | UINT32 | RW |
| 6083 $_h$ | VAR | profile_acceleration | UINT32 | RW |
| 6084 $_h$ | VAR | profile_deceleration | UINT32 | RW |
| 6085 $_h$ | VAR | quick_stop_deceleration | UINT32 | RW |
| 6086 $_h$ | VAR | motion_profile_type | INT16 | RW |
| 60A4-01 h | VAR | Profile_jerk1 | UINT32 | RW |

**target_position**

The object **target_position** determines the destination the servo controller moves to. The target position (**target_position**) is interpreted either as an absolute or relative position. This depends on bit 6 (**relative**) of the object **control word**.

| | |
|---|---|
| Index | 607A $_h$ |
| Name | target_ position |
| Object Code | VAR |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | position units |
| Value Range | -- |
| Default Value | 0 |

**profile_velocity**

The object **profile_velocity** specifies the speed that usually is reached during a positioning motion at the end of the acceleration ramp. The object **profile_velocity** is specified in **speed_units**.

| | |
|---|---|
| Index | 6081 $_h$ |
| Name | profile_velocity |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 0 |

**end_velocity**

The object **end_velocity** defines the speed at the target position (**target_position**). Usually this object has to be set to zero so that the controller stops when it reaches the target position. For gapless sequences of positionings a value unequal zero can be set.

| | |
|---|---|
| Index | 6082 h |
| Name | end_velocity |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | speed units |
| Value Range | -- |
| Default Value | 0 |

**profile_acceleration**

The object **profile_acceleration** determines the maximum acceleration used during a positioning motion. It is specified in user specific acceleration units (**acceleration_units)**.

| | |
|---|---|
| Index | 6083 h |
| Name | profile_acceleration |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | acceleration units |
| Value Range | -- |
| Default Value | 100000 R/10min/s |

**profile_deceleration**

The object **profile_deceleration** specifies the maximum deceleration used during a positioning motion. This object is specified in the same units as the object **profile_acceleration**

| | |
|---|---|
| Index | 6084 h |
| Name | profile_deceleration |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | acceleration units |
| Value Range | -- |
| Default Value | 100000 R/10min/s |

**quick_stop_deceleration**

The object **quick_stop_deceleration** determines the deceleration if a Quick Stop will be executed.

| | |
|---|---|
| Index | 6085 h |
| Name | quick_stop_deceleration |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | acceleration units |
| Value Range | -- |
| Default Value | 200000 R/10min/s |

**motion_profile_type**

The object **motion_profile_type** is used to select the kind of speed profile. At present only a linear trpezia profile(set as 0) and a stable S linear jerk profile are available(set as 2).

| | |
|---|---|
| Index | 6086 h |
| Name | motion_profile_type |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | YES |
| Units | -- |
| Value Range | 0 |
| Default Value | 0 |

**profile_jerk1**

profile_jerk1 is used to set the jerk of speed profile. The value is more smaller,the speed changing is more smooth.

| | |
|---|---|
| Index | 60A4 -01h |
| Name | profile_jerk1 |
| Object Code | VAR |
| Data Type | UINT32 |
| Access | RW |
| PDO Mapping | YES |
| Units | jerk units |
| Value Range | 1-20 |
| Default Value | 5pulse/(s*100μs*100μs) |

## 7.4.4 Function Description

When the speed profile is trapezia(motion_profile_type=0),two different ways to apply target positions to the servo controller are supported.

**Single setpoints**

After reaching the **target_position** the servo controller signals this status to the host by the bit **target_reached** (Bit 10 of **controlword**) and then receives a new setpoint. The servo controller stops at the **target_position** before starting a move to the next setpoint.

**Set of setpoints**

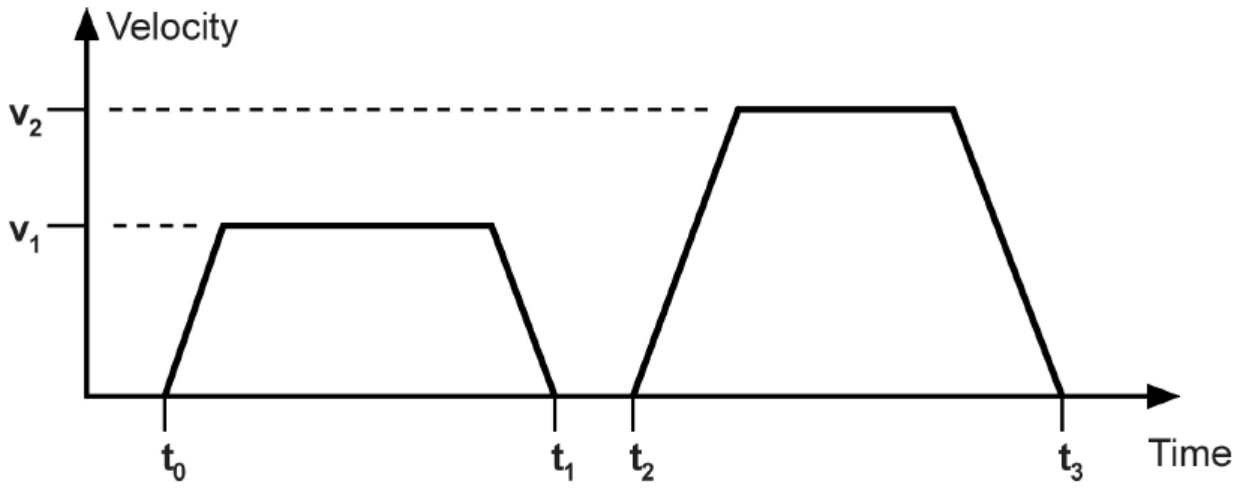
After reaching the **target_position** the servo controller immediately processes the next **target_position** which results in a move where the velocity of the drive normally is not reduced to zero after reaching a setpoint.

These Two methods are controlled by the bit4 and **bit5** in the object **controlword** and **set_point_acknowledge** in the object **statusword**. These bits are in a request-response relationship. So it is possible to prepare one positioning job while another job is still running.

Simple job positioning:

At first set NMT as Operational and control mode parameter (6061h) as 1.

1. At first the positioning data (**target_position:** 607A $_h$ **, profile_velocity, end_velocity** and **profile_acceleration**) are transferred to the servo controller.
2. The host can start the positioning motion by setting the bit4 (**new_set_point)** in the **controlword** as 1, bit5 (change_set_immediately) as 0 and bit6 as absolute or referential type according to target position type (absolute or referential).
3. This will be acknowledged by the servo controller by setting the bit **set_point_acknowledge** in the **statusword** when the positioning data has been copied into the internal buffer. Motion could be started now.
4. When the target is reached, drive will be acknowledged by bit 10 (target_reached) in status word. And then it will run gapless according to program or accept a new target position.




Gapless sequence of positioning job:

At first set NMT as Operational and control mode parameter (6061h) as 1.

1. At first the positioning data (**target_position:** 607A$_h$**, profile_velocity, end_velocity** and **profile_acceleration**) are transferred to the servo controller.
2. The host can start the positioning motion by setting the bit4 (**new_set_point)** in the **controlword** as 1, bit5 (change_set_immediately) as 0 and bit6 as absolute or referential type according to target position type (absolute or

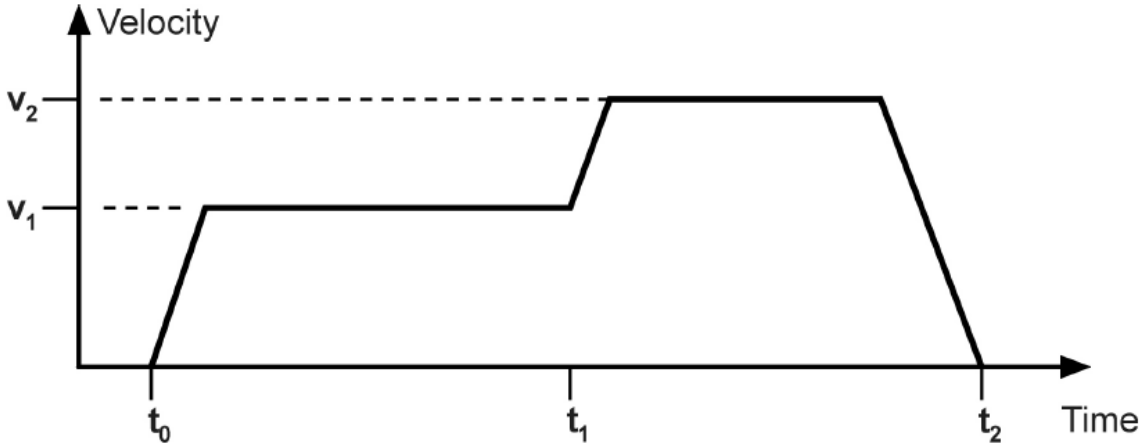
referential).

3. This will be acknowledged by the servo controller by setting the bit **set_point_acknowledge** in the **statusword** when the positioning data has been copied into the internal buffer. Motion could be started now.

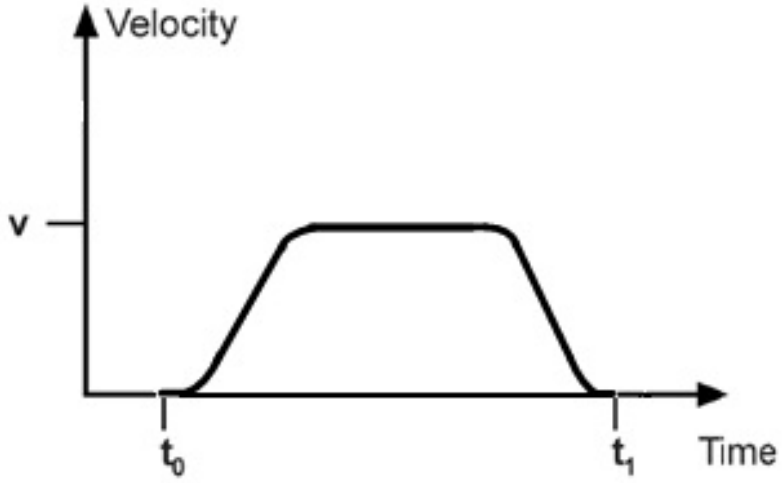
4.Second positioning data (**target_position:** 607A$_h$**, profile_velocity, end_velocity** and **profile_acceleration**) are transferred to the servo controller.

5. The host can start the positioning motion by setting the bit4 (**new_set_point)** in the **controlword** as 1, bit5 (change_set_immediately) as 0 and bit6 as absolute or referential type according to target position type (absolute or referential).

6. When the 1 target is reached driver will move forward to second target position. When the second target position is reached drive will be acknowledged by bit10 (target reached) in status word. And then it will be executed by program or accept another new target position.




When the speed profile is S (motion_profile_type=2), only single setpoints is available. 6083 h (profile_acceleration) limits max. acceleration. 6081h(profile_velocity) limits max.speed. 60A4-01 h (VAR Profile_jerk1) limits the jerk.now only symmetrical S linear is available.



# 7.5 Interpolation position mode

## 7.5.1 Control word of interpolation position mode

| 15 ~ 9 | 8 | 7 | 6 | 5 | 4 | 3 ~ 0 |
|---|---|---|---|---|---|---|
| * | Halt | * | * | * | Enable ip mode | * |

*: Please refer to the chapters ahead

| Name | Value | Description |
|---|---|---|
| Enable ip mode | 0 | Interpolated position mode inactive |
| | 1 | Interpolated position mode active |
| Halt | 0 | Execute the instruction of bit 4 |
| | 1 | Stop axle |

## 7.5.2 Status word of interpolation position mode

| 15 ~ 14 | 13 | 12 | 11 | 10 | 9 ~ 0 |
|---|---|---|---|---|---|
| * | * | ip mode active | * | Target reached | * |

*: Please refer to the chapters ahead

| Name | Value | Description |
|---|---|---|
| Target reached | 0 | Halt = 0: Position not (yet) reached<br>Halt = 1: Axle decelerates |
| | 1 | Halt = 0: Position reached<br>Halt = 1: Axle has velocity 0 |
| ip mode active | 0 | Interpolated position mode inactive |
| | 1 | Interpolated position mode active |

## 7.5.3 Parameters of position interpolation control

| Index | Object | Name | Type | Attr. |
|---|---|---|---|---|
| 60C0 h | VAR | Interpolation sub mode select | INT16 | RW |
| 60C1 h | ARRAY | Interpolation data record | INT32 | RW |
| 60C2 h | RECORD | Interpolation time period | | RW |

**Interpolation sub mode select**

Interpolation sub mode select is used to select the method of interpolation under IP control. Pronet servo drive only offers linear interpolation.

| | |
|---|---|
| Index | 60C0h |
| Name | Interpolation sub mode select |
| Object Code | VAR |
| Data Type | INT16 |
| Access | RW |
| PDO Mapping | NO |
| Value Range | 0 |
| Default Value | 0 |
| Comment | 0: Linear interpolation |

**Interpolation data record**

Interpolation data record is used to reserve interpolation potion data. Our servo drive's interpolation command only uses the first data whose subindex is 1.

| | |
|---|---|
| Index | 60C1h |
| Subindex | 0 |
| Object Code | ARRAY |
| Data Type | INT32 |
| Access | RO |
| PDO Mapping | YES |
| Value Range | INT8 |
| Default Value | 2 |
| Comment | number of entries |

| | |
|---|---|
| Index | 60C1h |
| Subindex | 1 |
| Object Code | ARRAY |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Value Range | INT32 |
| Default Value | 0 |
| Comment | the first parameter of ip function |

| | |
|---|---|
| Index | 60C1h |
| Subindex | 2 |
| Object Code | ARRAY |
| Data Type | INT32 |
| Access | RW |
| PDO Mapping | YES |
| Value Range | INT32 |
| Default Value | 0 |
| Comment | The second parameter of ip function |

**Interpolation time period**

Interpolation time period is used to reserve the time data of interpolation position.

| | |
|---|---|
| Index | 60C2h |
| Object Code | RECORD |
| Data Type | Interpolation time period record (0080h) |
| Category | Conditional: mandatory if ip, csp, csv or cst mode is supported |

| | |
|---|---|
| Index | 60C2h |
| Subindex | 0 |
| Object Code | RECORD |
| Data Type | UINT8 |
| Access | C |
| PDO Mapping | NO |
| Value Range | 02 |
| Default Value | 02 |
| Comment | Highest sub-index supported |

| | |
|---|---|
| Index | 60C2h |
| Subindex | 01 |
| Object Code | RECORD |
| Data Type | UINT8 |
| Access | RW |
| PDO Mapping | YES |
| Value Range | UINT8 |
| Default Value | 01 |
| Comment | Interpolation time period value |

| | |
|---|---|
| Index | 60C2h |
| Subindex | 02 |
| Object Code | RECORD |
| Data Type | INT8 |
| Access | RW |
| PDO Mapping | YES |
| Value Range | -128 to +63 |
| Default Value | -3 |
| Comment | Interpolation time index |

## 7.5.4 Function description

Some hints:

1. In our servo drive, there is no buffer for position data so in IP control, all the position data needs to be updated by the controller. To achieve synchronization, controllers need to send the updated position at first and then use SYNC signal to make all the servo drive receive the synchronization information. After receiving the synchronization information, servo drive will synchronize its internal clock. Please notice that the sync period should be not bigger than interpolation cycle period in order to keep the updating of interpolation data.
2. In IP mode, the host should at first set the servo's PDO receiving method into sync mode ( Use SYNC frame to receive and send synchronization information). Because SYNC is broad casted, every servo drive will only update PDO data after receiving this signal.
3. Before SYNC is sent, we need host to send position data Xi and control word to the servo drive.
4. When there is data delay, servo drive will use the last sync date to do interpolation.
5. After one sync period, if there is no further data updating, interpolation cycle overtime alarm （A 69）will happen. And then servo drive will stop.

Recommended RPDO configuration:

When you use only one RPDO,

| | |
|---|---|
| Control word(index:6040h,subindex:0h) | 32bit position reference (index:60C1h,subindex:01h) |

When you use two RPDO,

| |
|---|
| Control word(index:6040h,subindex:0h) |

| |
|---|
| 32bit position reference (index:60C1h,subindex:01h) |

Configuration process:

1. Configure PDO. (RPDO1 is configured as index: 6040h, subindex: 0h, RPDO2 is configured as index 60c1h, subindex: 1h)

2. Set interpolation cycle time 2105h and 60C2, the unit is micro send (us). Please notice that both values need to be configured. For example, if the cycle time is 2ms, you need to set 2105h as 2000 and 60c2:01 as 2, 60c2:02 as -3.

3. Set sync cycle time (1006h), the unit is micro send (us)

4. Set PDO as Sync mode (Set the object dictionary (index: 1400h, subindex: 02h) as 1. Set object dictionary (index: 1401h, subindex: 02h) as 1). If sending PDO needs to be in sync mode as well, we need to set object dictionary (index: 1800h, subindex: 02h) as 1 and (index: 6060h, subindex: 0h) as 1 as well.

5. NMT starts node.

# Chapter 8 Parameters of the CAN interface

| Parameter | Name and discription | Reboot required | Available for which control method | Functions and content |
|---|---|---|---|---|
| Pn006 | **Hexadecimal** | required | ALL | **Pn006.0 Bus type selection**<br>[0]No bus<br>[1]PROFIBUS-DP V0/V1<br>[2]PROFIBUS-DP V2<br>[3] CANopen<br>**Pn006.1 Reserved**<br>**Pn006.2 Low-frequency vibration suppression switch**<br>[0]Low-frequency vibration suppression function disabled<br>[1]Low-frequency vibration suppression function enabled<br>**Pn006.3 Reference input filter for open collector signal**<br>[0] when pulse is difference input, servo receiving pulse frequency ≤4M<br>[1] when pulse is difference input, servo receiving pulse frequency ≤650K<br>[2] when pulse is difference input, servo receiving pulse frequency ≤150K |
| Pn703 | **Hexadecimal** | required | ALL | **Pn703.0** CANopen **baud rate**<br>[0] 50Kbps<br>[1] 100Kbps<br>[2] 125Kbps<br>[3] 250Kbps<br>[4] 500Kbps<br>[5] 1Mbps<br>**Pn703.1 Reserved**<br>**Pn703.2 Reserved**<br>**Pn703.3 Reserved** |
| Pn704 | **Axis address** | required | ALL | CANopen axis address |

# Chapter 9 CAN communication example

The entire test below is based on two conditions …

1. Communication has been established correctly.
2. The address of the servo drive is 1.

## 9.1 SDO configuration

SDO operation is to read and write parameters (06001→ host sends 0581-→slave sends)

Address: 0x3022（Pn034）. Write 1000. And then read this parameter.

Activate the downloading process: 2B，3022，00，FC18

That is …

601 2B 22 30 00 18 FC 00 00

The servo drive should respond 60，3022，00，00，00，00，00

That is 581 60 22 30 00 00 00 00 00

Activate the uploading: 40，3022，00，0000

That is 601 40 22 30 00 00 00 00 00

The servo drive needs to respond: 43，3022，00，FC18

That is: 581 43 22 30 00 18 FC 00 00

## 9.2 PDO Configuration

// pulse，Speed 0.1rpm

Example: To configure two RPDO, one of which is 6040h and the other are 607A and 6081h)

RPDO MAPPing

601 2F 00 16 00 00 00 00 00 //RPDO1 stop

first RPDO 201

601 23 00 16 01 10 00 40 60 //6040h

601 2F 00 16 00 01 00 00 00 // RPDO1 enable

601 2F 01 16 00 00 00 00 00 //RPDO2 stop

Second RPDO 301

601 23 01 16 01 20 00 7A 60 //607Ah and 6081h

601 23 01 16 02 20 00 81 60

601 2F 01 16 00 02 00 00 00// RPDO2 enable

And then set the transmit PDO as SYNC or Timing method. The default setting is Time method.

After configuring the PDO, if you need to activate the configuration, you need to reset the communication.
NMT is OPERATIONAL: 00 01 01//（the first “01” is the start node instruction,the second “01” is the number of the node）

Attention：
1.Before configuration, please stop PDO. For example, Cleaning the value with index 1600h and sub-index 00, cleaning the value to 0 is necessary). After configuration, please set a correct number of PDO( For example, set the value with index 1600h and sub-index 00 as 1) to activate the PDO.
2. Please pay attention to the data length and number. Wrong setting will lead to wrong configuration.

## 9.3 Profile Position Mode

At first, please configure PDO according to the example above and activate the communication.
And then, please set the control mode.
601 2F 60 60 00 01 00 00 00//set 6060h as 1（position contrl is PP）
And then, set status machine
601 2B 40 60 00 06 00 00 00//set 6040h as 6
601 2B 40 60 00 07 00 00 00 //set 6040h as 7
601 2B 40 60 00 0F 00 00 00 //set 6040h as F，servo-on；

And then, send data by PDO
Let servo motor rotate for 5 revolutions ( Set PDO1 as 6040(status word), PDO2 as 607A(position pulse number) and 6081(velocity, unit as much as 0.1rpm)

Send RPDO2 The data is as below …
301 50 C3 00 00 2C 01 00 00(50000,300)// 50 C3 00 00 is position data，that is, 50000 pulses；2C 01 00 00 is speed，that is, 30rpm；

Send RPDO1 as below
1、201 0F 00 //；Clear the bit4 of 6040 as 0.
2、201 1F 00 // Clear the bit4 of 6040 as 1 and servo motor is operating under absolute position; Motor runs.
3、201 0F 00 //Clear the bit4 of 6040.
4、201 5F 00 // Clear the bit4 of 6040 as 1. The servo motor runs under incremental position.
5、201 0F 00 //Clear bit4 of 6040 as 0.

Attention:
1) The servo drive is using ↑of 6040’s bit 4 to accept new position order. So after every single operation, the bit needs to be cleared. Host needs to check bit12 of status word 6040 in the servo drive to decide whether or not to give new data to servo systems. When status word 6041 in the servo drives 0, it means the servo drive is ready for new data and order. If the value is 1, the order won’t be executed even if there is data for the servo drive to receive.
2) In absolute approach, continuous position updating is required.

If you want to change the operating distance, you need to send RPDO2 again.
RPDO2:
301 B0 3C FF FF 2C 01 00 00 (-50000,-300)//That is, -50000 pulses; 30rpm.

# 9.4 Interplate Position Mode

At first, configure PDO
/receive 2 PDO by default：RPDO1：6040 RPDO2：60C1,sub01
// Send two PDO by default：TPDO1：6041 TPDO2：6064/606C
// pulse，Velocity 0.1rpm

Configure 2 RPDO，RPDO1：6040h　RPDO2：60C1h, sub01
RPDO MAPPing
601 2F 00 16 00 00 00 00 00 //RPDO1 stop
first RPDO 201
601 23 00 16 01 10 00 40 60 //6040h
601 2F 00 16 00 01 00 00 00 // RPDO1 enable

601 2F 01 16 00 00 00 00 00 //RPDO2 stop
Second RPDO 301
601 23 01 16 01 20 01 C1 60 //60C1h,sub01
601 2F 01 16 00 01 00 00 00// RPDO2 enable

Configure 2 TPDO，TPDO1：6041h TPDO2：6064h/606Ch
RPDO MAPPing
601 2F 00 1A 00 00 00 00 00 //TPDO1 stop
first RPDO 181
601 23 00 1A 01 10 00 41 60 //6041h
601 2F 00 1A 00 01 00 00 00 // TPDO1 enable

601 2F 01 1A 00 00 00 00 00 //RPDO2 stop
Second RPDO 281
601 23 01 1A 01 20 00 64 60 //6064h and 606Ch
601 23 01 1A 02 20 00 6C 60 //
601 2F 01 1A 00 02 00 00 00// TPDO2 enable

Set Sync time.
601 23 06 10 00 E8 03 00 00 //1006h----->1000us

Set interpolate time
601 2F C2 60 01 10 00 00 00　//60C2h----->1ms
601 2F C2 60 02 FD 00 00 00　//

Configure the PDO receiving and sending both by the means of the sync step and sync frame.
Set 1400h
601 2F 00 14 02 01 00 00 00 //1400----□SYNC
Set 1401h
601 2F 01 14 02 01 00 00 00 //1401----□SYNC

Set 1800h

601 2F 00 18 02 01 00 00 00 //1800----□SYNC

Set 1801h

601 2F 01 18 02 01 00 00 00 //1801----□SYNC

Reset the communication to active dynamic PDO configuration.

00 82 01 //reset communication

Set control mode

601 2F 60 60 00 07 00 00 00// ( IP position control)

And then, set the status machine

601 2B 40 60 00 06 00 00 00//Set 6040h as 6

601 2B 40 60 00 07 00 00 00 //Set 6040h as 7

601 2B 40 60 00 0F 00 00 00 // Set 6040h as F to servo on.

Activate the communicaiton

00 01 01

The host sends signals by the period of 1000us.

301 10 00 00 00 //16 pulses

201 1F 00 //IP

80 periodical sending

## 9.5 Homing

Set the control mode as homing control.

601 2F 60 60 00 06 00 00 00// Set the control mode as homing control.

601 2F 98 60 00 04 00 00 00//Use the fourth way to set the homing mode.

Set the status machine

601 2B 40 60 00 06 00 00 00

601 2B 40 60 00 07 00 00 00

601 2B 40 60 00 0F 00 00 00 //Servo On

Send data through PDO. ( Set PDO1 as 6040( status word). Set PDO2 as 607A( Position pulse number) and 6081. ( Speed, unit 0.1rpm)

Set the homing method as 10rpm.

601 23 99 60 02 64 00 00 00

Homing is started.

201 1F 00

Cancel homing.

201 0F 00

# Chapter 10 Other function

## 10.1 Bus input

In some case, the switch (homing and limit switch) is committed by upper computer not servo drive.the signal is committed by object 60FE-01h(Physical outputs).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23…0 |
|---|---|---|---|---|---|---|---|---|
| CN1_42 | CN1_41 | CN1_40 | CN1_39 | CN1_17 | CN1_16 | CN1_15 | CN1_14 | reserved |

8 high- bit of the object corresponding input signal terminal connector(CN1). The function of the terminals is formed by Pn509/510 or inversing Pn516/517. Enabled bus commit by Pn512/Pn513 for the bus commit bits. Refer to <ProNet Series AC Servo User's Manual>.

| **Index** | 60FE $_{h}$ |
|---|---|
| **Name** | Digital outputs |
| **Object Code** | ARRAY |
| **No. of Elements** | 2 |
| **Data Type** | UINT32 |

| **Sub-Index** | 01 $_{h}$ |
|---|---|
| **Name** | Physical outputs |
| **Object Code** | VAR |
| **Data Type** | UINT32 |
| **Access** | RW |
| **PDO Mapping** | YES |
| **Default Value** | 0 |

| **Sub-Index** | 02 $_{h}$ |
|---|---|
| **Name** | Bit mask |
| **Object Code** | VAR |
| **Data Type** | UINT32 |
| **Access** | RW |
| **PDO Mapping** | YES |
| **Default Value** | 0 |

The host can read the object 60FDh(Digital Inputs) to monitor the switching inputs of the servo.

| 31-26 | 2 | 1 | 0 |
|---|---|---|---|
| reserved | home switch | positive limit switch | negative limit switch |

# 10.2 Dummy object

To a better performance and save bandwidth of the bus ,the host can post data to the different slaves by the same PDO. COB-ID of slave's RPDO should be set as the same value,and dummy object should be mapped to slave's RPDO. Sometimes need to turn off some needless TPDO.

Example of mapping the first RPDO of two nodes:

Node1

0x1600-1 = 60C1 00 10H // Interpolated position

0x1600-2 = 0007 00 10H // Dummy object 32 bit

0x1600-0 = 2H

Node 2

0x1600-1 = 0007 00 10H // Dummy object 32 bit

0x1600-2 = 60C1 00 10H // Interpolated position

0x1600-0 = 2H

Host controller can use one PDO，and map the two slaves'interpolated position to the first and the second objects at the same time.

Example of programming COB-ID: revise COB-ID of Node 2 the first RPDO from 0x202 to 0x201

0x1400-1 = 80000201H //Restrict PDO，and write new COB-ID

0x1400-1 = 00000201H // Enabled PDO

Example of closing TPDO: restrict the second TPDO of Node 2 from transmitting （default COB-ID is 0x282）

0x1801-1 = 80000282H

# Appendix Object dictionary

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | PP | PV | HM | |
| 2 | -- | VAR | od_integer8 | INT8 | RW | YES | ● | | | | |
| 3 | -- | VAR | od_integer16 | INT16 | RW | YES | ● | | | | |
| 4 | -- | VAR | od_integer32 | INT32 | RW | YES | ● | | | | |
| 5 | -- | VAR | od_unsigned8 | UINT8 | RW | YES | ● | | | | |
| 6 | -- | VAR | od_unsigned16 | UINT16 | RW | YES | ● | | | | |
| 7 | -- | VAR | od_unsigned32 | UINT32 | RW | YES | ● | | | | |
| 1000 | -- | VAR | device_type | UINT32 | RO | NO | ● | | | | |
| 1001 | -- | VAR | error_register | UINT8 | RO | NO | ● | | | | |
| 1003 | -- | VAR | pre_defined_error_field | UINT8 | RW | NO | ● | | | | |
| 1005 | -- | VAR | cob_id_sync | UINT32 | RW | NO | ● | | | | |
| 1006 | -- | VAR | communication_cycle_period | UINT32 | RW | NO | ● | | | | |
| 1007 | -- | VAR | synchronous_window_length | UINT32 | RW | NO | ● | | | | |
| 1014 | -- | VAR | cob_id_emergency_message | UINT32 | RW | NO | ● | | | | |
| 1016 | -- | ARRAY | consumer_heartbeat_time | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | consumer_heartbeat_time1 | UINT32 | RW | NO | ● | | | | |
| 1017 | | VAR | producer_heartbeat_time | UINT16 | RW | NO | ● | | | | |
| 1018 | -- | RECORD | identity_object | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | vendor_id | UINT32 | RO | NO | ● | | | | |
| | 2 | | product_code | UINT32 | RO | NO | ● | | | | |
| | 3 | | revision_number | UINT32 | RO | NO | ● | | | | |
| | 4 | | serial_number | UINT32 | RO | NO | ● | | | | |
| 1029 | -- | ARRAY | error_behaviour | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | communication_error | UINT8 | RW | NO | ● | | | | |
| 1200 | -- | RECORD | server_sdo_parameter | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | cob_id_client_server | UINT32 | RO | NO | ● | | | | |
| | 2 | | cob_id_server_client | UINT32 | RO | NO | ● | | | | |
| 1400 | -- | RECORD | receive_pdo_parameter_rpdo1 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_rpdo1 | UINT8 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_rpdo1 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_rpdo1 | UINT8 | RW | NO | ● | | | | |
| 1401 | -- | RECORD | receive_pdo_parameter_rpdo2 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_rpdo2 | UINT8 | RO | NO | ● | | | | |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | PP | PV | HM | |
| | 1 | | cob_id_used_by_pdo_rpdo2 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_rpdo2 | UINT8 | RW | NO | ● | | | | |
| 1402 | -- | RECORD | receive_pdo_parameter_rpdo3 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_rpdo3 | UINT8 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_rpdo3 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_rpdo3 | UINT8 | RW | NO | ● | | | | |
| 1403 | -- | RECORD | receive_pdo_parameter_rpdo4 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_rpdo4 | UINT8 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_rpdo4 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_rpdo4 | UINT8 | RW | NO | ● | | | | |
| 1600 | -- | RECORD | receive_pdo_mapping_rpdo1 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_rpdo1 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_rpdo1 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_rpdo1 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_rpdo1 | UINT32 | RW | NO | ● | | | | |
| 1601 | -- | RECORD | receive_pdo_mapping_rpdo2 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_rpdo2 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_rpdo2 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_rpdo2 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_rpdo2 | UINT32 | RW | NO | ● | | | | |
| 1602 | -- | RECORD | receive_pdo_mapping_rpdo3 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_rpdo3 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_rpdo3 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_rpdo3 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_rpdo3 | UINT32 | RW | NO | ● | | | | |
| 1603 | -- | RECORD | receive_pdo_mapping_rpdo4 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_rpdo4 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_rpdo4 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_rpdo4 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_rpdo4 | UINT32 | RW | NO | ● | | | | |
| 1800 | -- | RECORD | transmit_pdo_parameter_tpdo1 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_tpdo1 | UINT32 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_tpdo1 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_tpdo1 | UINT8 | RW | NO | ● | | | | |
| | 3 | | inhibit_time_tpdo1 | UINT16 | RW | NO | ● | | | | |
| | 5 | | event_timer_tpdo1 | UINT16 | RW | NO | ● | | | | |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | PP | PV | HM | |
| 1801 | -- | RECORD | transmit_pdo_parameter_tpdo2 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_tpdo2 | UINT32 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_tpdo2 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_tpdo2 | UINT8 | RW | NO | ● | | | | |
| | 3 | | inhibit_time_tpdo2 | UINT16 | RW | NO | ● | | | | |
| | 5 | | event_timer_tpdo2 | UINT16 | RW | NO | ● | | | | |
| 1802 | -- | RECORD | transmit_pdo_parameter_tpdo3 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_tpdo3 | UINT32 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_tpdo3 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_tpdo3 | UINT8 | RW | NO | ● | | | | |
| | 3 | | inhibit_time_tpdo3 | UINT16 | RW | NO | ● | | | | |
| | 5 | | event_timer_tpdo3 | UINT16 | RW | NO | ● | | | | |
| 1803 | -- | RECORD | transmit_pdo_parameter_tpdo4 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries_tpdo4 | UINT32 | RO | NO | ● | | | | |
| | 1 | | cob_id_used_by_pdo_tpdo4 | UINT32 | RO | NO | ● | | | | |
| | 2 | | transmission_type_tpdo4 | UINT8 | RW | NO | ● | | | | |
| | 3 | | inhibit_time_tpdo4 | UINT16 | RW | NO | ● | | | | |
| | 5 | | event_timer_tpdo4 | UINT16 | RW | NO | ● | | | | |
| 1A00 | -- | RECORD | transmit_pdo_mapping_tpdo1 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_tpdo1 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_tpdo1 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_tpdo1 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_tpdo1 | UINT32 | RW | NO | ● | | | | |
| 1A01 | -- | RECORD | transmit_pdo_mapping_tpdo2 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_tpdo2 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_tpdo2 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_tpdo2 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_tpdo2 | UINT32 | RW | NO | ● | | | | |
| 1A02 | -- | RECORD | transmit_pdo_mapping_tpdo3 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_tpdo3 | UINT32 | RW | NO | ● | | | | |
| | 2 | | second_mapped_object_tpdo3 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_tpdo3 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_tpdo3 | UINT32 | RW | NO | ● | | | | |
| 1A03 | -- | RECORD | transmit_pdo_mapping_tpdo4 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | first_mapped_object_tpdo4 | UINT32 | RW | NO | ● | | | | |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | PP | PV | HM | |
| | 2 | | second_mapped_object_tpdo4 | UINT32 | RW | NO | ● | | | | |
| | 3 | | third_mapped_object_tpdo4 | UINT32 | RW | NO | ● | | | | |
| | 4 | | fourth_mapped_object_tpdo4 | UINT32 | RW | NO | ● | | | | |
| 2000 | -- | RECORD | mask_tpdo1 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | mask1_tpdo1 | UINT32 | RW | NO | ● | | | | |
| | 2 | | mask2_tpdo1 | UINT32 | RW | NO | ● | | | | |
| 2001 | -- | RECORD | mask_tpdo2 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | mask1_tpdo2 | UINT32 | RW | NO | ● | | | | |
| | 2 | | mask2_tpdo2 | UINT32 | RW | NO | ● | | | | |
| 2002 | -- | RECORD | mask_tpdo3 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | mask1_tpdo3 | UINT32 | RW | NO | ● | | | | |
| | 2 | | mask2_tpdo3 | UINT32 | RW | NO | ● | | | | |
| 2003 | -- | RECORD | mask_tpdo4 | -- | -- | -- | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | ● | | | | |
| | 1 | | mask1_tpdo4 | UINT32 | RW | NO | ● | | | | |
| | 2 | | mask2_tpdo4 | UINT32 | RW | NO | ● | | | | |
| 2105 | 0 | VAR | sync_time_period | UINT32 | RW | NO | ● | | | | |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | IP | PP | PV | HM | |
| 3000 | -- | VAR | Pn000 | UINT16 | RW | NO | ● | | | | | -- |
| 3001 | -- | VAR | Pn001 | UINT16 | RW | NO | ● | | | | | -- |
| 3002 | -- | VAR | Pn002 | UINT16 | RW | NO | ● | | | | | -- |
| 3003 | -- | VAR | Pn003 | UINT16 | RW | NO | ● | | | | | -- |
| 3004 | -- | VAR | Pn004 | UINT16 | RW | NO | ● | | | | | -- |
| 3005 | -- | VAR | Pn005 | UINT16 | RW | NO | ● | | | | | -- |
| 3006 | -- | VAR | Pn006 | UINT16 | RW | NO | ● | | | | | -- |
| 3007 | -- | VAR | Pn007 | UINT16 | RW | NO | ● | | | | | -- |
| 3010 | -- | VAR | Pn100 | UINT16 | RW | NO | ● | | | | | -- |
| 3011 | -- | VAR | Pn101 | UINT16 | RW | NO | ● | | | | | -- |
| 3012 | -- | VAR | Pn102 | UINT16 | RW | NO | ● | | | | | Hz |
| 3013 | -- | VAR | Pn103 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 3014 | -- | VAR | Pn104 | UINT16 | RW | NO | ● | | | | | 1/s |
| 3015 | -- | VAR | Pn105 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 3016 | -- | VAR | Pn106 | UINT16 | RW | NO | ● | | | | | -- |
| 3017 | -- | VAR | Pn107 | UINT16 | RW | NO | ● | | | | | Hz |
| 3018 | -- | VAR | Pn108 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 3019 | -- | VAR | Pn109 | UINT16 | RW | NO | ● | | | | | Hz |
| 301A | -- | VAR | Pn110 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 301B | -- | VAR | Pn111 | UINT16 | RW | NO | ● | | | | | r/min |
| 301C | -- | VAR | Pn112 | UINT16 | RW | NO | ● | | | | | % |
| 301D | -- | VAR | Pn113 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 301E | -- | VAR | Pn114 | UINT16 | RW | NO | ● | | | | | % |
| 301F | -- | VAR | Pn115 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 3020 | -- | VAR | Pn116 | UINT16 | RW | NO | ● | | | | | -- |
| 3021 | -- | VAR | Pn117 | UINT16 | RW | NO | ● | | | | | % |
| 3022 | -- | VAR | Pn118 | UINT16 | RW | NO | ● | | | | | -- |
| 3023 | -- | VAR | Pn119 | UINT16 | RW | NO | ● | | | | | -- |
| 3024 | -- | VAR | Pn120 | UINT16 | RW | NO | ● | | | | | -- |
| 3025 | -- | VAR | Pn121 | UINT16 | RW | NO | ● | | | | | -- |
| 3026 | -- | VAR | Pn122 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 3027 | -- | VAR | Pn123 | UINT16 | RW | NO | ● | | | | | -- |
| 3028 | -- | VAR | Pn124 | UINT16 | RW | NO | ● | | | | | -- |
| 3029 | -- | VAR | Pn125 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 302A | -- | VAR | Pn126 | UINT16 | RW | NO | ● | | | | | -- |
| 302B | -- | VAR | Pn127 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 302C | -- | VAR | Pn128 | UINT16 | RW | NO | ● | | | | | 0.1% |
| 302D | -- | VAR | Pn129 | UINT16 | RW | NO | ● | | | | | r/min |
| 302E | -- | VAR | Pn130 | UINT16 | RW | NO | ● | | | | | 0.1% |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | IP | PP | PV | HM | |
| 302F | -- | VAR | Pn131 | UINT16 | RW | NO | ● | | | | | r/min |
| 3030 | -- | VAR | Pn132 | UINT16 | RW | NO | ● | | | | | 0.1%/1000rpm |
| 3055 | -- | VAR | Pn305 | UINT16 | RW | NO | ● | | | | | rpm |
| 3056 | -- | VAR | Pn306 | UINT16 | RW | NO | ● | | | | | ms |
| 3057 | -- | VAR | Pn307 | UINT16 | RW | NO | ● | | | | | ms |
| 3058 | -- | VAR | Pn308 | UINT16 | RW | NO | ● | | | | | ms |
| 3059 | -- | VAR | Pn309 | UINT16 | RW | NO | ● | | | | | ms |
| 305A | -- | VAR | Pn310 | UINT16 | RW | NO | ● | | | | | -- |
| 305B | -- | VAR | Pn311 | UINT16 | RW | NO | ● | | | | | -- |
| 3068 | -- | VAR | Pn401 | UINT16 | RW | NO | ● | | | | | % |
| 3069 | -- | VAR | Pn402 | UINT16 | RW | NO | ● | | | | | % |
| 306A | -- | VAR | Pn403 | UINT16 | RW | NO | ● | | | | | % |
| 306B | -- | VAR | Pn404 | UINT16 | RW | NO | ● | | | | | % |
| 306C | -- | VAR | Pn405 | UINT16 | RW | NO | ● | | | | | % |
| 306D | -- | VAR | Pn406 | UINT16 | RW | NO | ● | | | | | rpm |
| 306E | -- | VAR | Pn407 | UINT16 | RW | NO | ● | | | | | HZ |
| 306F | -- | VAR | Pn408 | UINT16 | RW | NO | ● | | | | | -- |
| 3070 | -- | VAR | Pn409 | UINT16 | RW | NO | ● | | | | | HZ |
| 3071 | -- | VAR | Pn410 | UINT16 | RW | NO | ● | | | | | -- |
| 3072 | -- | VAR | Pn411 | UINT16 | RW | NO | ● | | | | | 0.1 HZ |
| 3073 | -- | VAR | Pn412 | UINT16 | RW | NO | ● | | | | | -- |
| 3074 | -- | VAR | Pn413 | UINT16 | RW | NO | ● | | | | | 0.1ms |
| 3075 | -- | VAR | Pn414 | UINT16 | RW | NO | ● | | | | | rpm |
| 3078 | -- | VAR | Pn500 | UINT16 | RW | NO | ● | | | | | pulse |
| 3079 | -- | VAR | Pn501 | UINT16 | RW | NO | ● | | | | | rpm |
| 307A | -- | VAR | Pn502 | UINT16 | RW | NO | ● | | | | | rpm |
| 307B | -- | VAR | Pn503 | UINT16 | RW | NO | ● | | | | | rpm |
| 307C | -- | VAR | Pn504 | UINT16 | RW | NO | ● | | | | | 256pulse |
| 307D | -- | VAR | Pn505 | INT16 | RW | NO | ● | | | | | ms |
| 307E | -- | VAR | Pn506 | UINT16 | RW | NO | ● | | | | | 10ms |
| 307F | -- | VAR | Pn507 | UINT16 | RW | NO | ● | | | | | rpm |
| 3080 | -- | VAR | Pn508 | UINT16 | RW | NO | ● | | | | | 10ms |
| 3081 | -- | VAR | Pn509 | UINT16 | RW | NO | ● | | | | | -- |
| 3082 | -- | VAR | Pn510 | UINT16 | RW | NO | ● | | | | | -- |
| 3083 | -- | VAR | Pn511 | UINT16 | RW | NO | ● | | | | | -- |
| 3084 | -- | VAR | Pn512 | UINT16 | RW | NO | ● | | | | | -- |
| 3085 | -- | VAR | Pn513 | UINT16 | RW | NO | ● | | | | | -- |
| 3086 | -- | VAR | Pn514 | UINT16 | RW | NO | ● | | | | | 0.2ms |
| 3088 | -- | VAR | Pn516 | UINT16 | RW | NO | ● | | | | | -- |
| 3089 | -- | VAR | Pn517 | UINT16 | RW | NO | ● | | | | | -- |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | IP | PP | PV | HM | |
| 3091 | -- | VAR | Pn525 | UINT16 | RW | NO | ● | | | | | % |
| 30FC | -- | VAR | Pn700 | UINT16 | RW | NO | ● | | | | | -- |
| 30FD | -- | VAR | Pn701 | UINT16 | RW | NO | ● | | | | | -- |
| 30FF | -- | VAR | Pn703 | UINT16 | RW | NO | ● | | | | | -- |
| 3100 | -- | VAR | Pn704 | UINT16 | RW | NO | ● | | | | | -- |
| 3200 | -- | VAR | Fn000-0 | UINT16 | RO | NO | ● | | | | | -- |
| 3201 | -- | VAR | Fn000-1 | UINT16 | RO | NO | ● | | | | | -- |
| 3202 | -- | VAR | Fn000-2 | UINT16 | RO | NO | ● | | | | | -- |
| 3203 | -- | VAR | Fn000-3 | UINT16 | RO | NO | ● | | | | | -- |
| 3204 | -- | VAR | Fn000-4 | UINT16 | RO | NO | ● | | | | | -- |
| 3205 | -- | VAR | Fn000-5 | UINT16 | RO | NO | ● | | | | | -- |
| 3206 | -- | VAR | Fn000-6 | UINT16 | RO | NO | ● | | | | | -- |
| 3207 | -- | VAR | Fn000-7 | UINT16 | RO | NO | ● | | | | | -- |
| 3208 | -- | VAR | Fn000-8 | UINT16 | RO | NO | ● | | | | | -- |
| 3209 | -- | VAR | Fn000-9 | UINT16 | RO | NO | ● | | | | | -- |
| 320A | -- | VAR | Un000 | UINT16 | RO | NO | ● | | | | | r/min |
| 320B | -- | VAR | Un001 | UINT16 | RO | NO | ● | | | | | r/min |
| 320C | -- | VAR | Un002 | UINT16 | RO | NO | ● | | | | | % |
| 320D | -- | VAR | Un003 | UINT16 | RO | NO | ● | | | | | % |
| 320E | -- | VAR | Un004 | UINT16 | RO | NO | ● | | | | | pulse |
| 320F | -- | VAR | Un005 | UINT16 | RO | NO | ● | | | | | -- |
| 3210 | -- | VAR | Un006 | UINT16 | RO | NO | ● | | | | | -- |
| 3211 | -- | VAR | Un007 | UINT16 | RO | NO | ● | | | | | -- |
| 3212 | -- | VAR | Un008 | UINT16 | RO | NO | ● | | | | | 1kHZ |
| 3213 | -- | VAR | Un009 | UINT16 | RO | NO | ● | | | | | pulse |
| 3214 | -- | VAR | Un010 | UINT16 | RO | NO | ● | | | | | -- |
| 3215 | -- | VAR | Un011 | UINT16 | RO | NO | ● | | | | | -- |
| 3216 | -- | VAR | Un012 | UINT16 | RO | NO | ● | | | | | -- |
| 3217 | -- | VAR | Un013 | UINT16 | RO | NO | ● | | | | | -- |
| 3218 | -- | VAR | Un014 | UINT16 | RO | NO | ● | | | | | -- |
| 321E | -- | VAR | DSP-Edition | UINT16 | RO | NO | ● | | | | | -- |
| 3300 | -- | VAR | rotates | UINT16 | RO | NO | ● | | | | | -- |
| 3301 | -- | VAR | singlePos | UINT16 | RO | NO | ● | | | | | -- |

| Index | Subindex | Object | Name | Type | Attr. | PDO | Support | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | IP | PP | PV | HM | |
| 603F | -- | VAR | error_code | UINT16 | RO | YES | ● | | | | | -- |
| 6040 | -- | VAR | controlword | UINT16 | RW | YES | ● | | | | | -- |
| 6041 | -- | VAR | statusword | UINT16 | RO | YES | ● | | | | | -- |
| 605A | -- | VAR | quick_stop_option_code | INT16 | RW | NO | ● | | | | | -- |
| 605B | -- | VAR | shutdown_option_code | INT16 | RW | NO | ● | | | | | -- |
| 605C | -- | VAR | disable_operation_option_code | INT16 | RW | NO | ● | | | | | -- |
| 605D | -- | VAR | stop_option_code | INT16 | RW | NO | ● | | | | | -- |
| 605E | -- | VAR | fault_reaction_option_code | INT16 | RW | NO | ● | | | | | -- |
| 6060 | -- | VAR | modes_of_operation | INT8 | RW | YES | ● | | | | | -- |
| 6061 | -- | VAR | modes_of_operation_display | INT8 | RO | YES | ● | | | | | -- |
| 6062 | -- | VAR | position_demand_value | INT32 | RO | YES | | | ● | | | position units |
| 6063 | -- | VAR | position_actual_value* | INT32 | RO | YES | | | ● | | | inc |
| 6064 | -- | VAR | position_actual_value | INT32 | RO | YES | | ● | ● | | | position units |
| 6065 | -- | VAR | following_error_window | UINT32 | RW | YES | | | ● | | | position units |
| 6066 | -- | VAR | following_error_time_out | UINT16 | RW | YES | | | ● | | | ms |
| 6067 | -- | VAR | position_window | UINT32 | RW | YES | | | ● | | | position units |
| 6068 | -- | VAR | position_window_time | UINT16 | RW | YES | | | ● | | | ms |
| 6069 | -- | VAR | velocity_sensor_actual_value | UINT32 | RO | YES | | | | ● | | speed units |
| 606B | -- | VAR | velocity_demand_value | INT32 | RO | YES | | | | ● | | speed units |
| 606C | -- | VAR | velocity_actual_value | INT32 | RO | YES | | | | ● | | speed units |
| 606D | -- | VAR | velocity_window | UINT16 | RW | YES | | | | ● | | speed units |
| 606E | -- | VAR | velocity_window_time | UINT16 | RW | YES | | | | ● | | ms |
| 606F | -- | VAR | velocity_threshold | UINT16 | RW | YES | | | | ● | | speed units |
| 6070 | -- | VAR | velocity_threshold_time | UINT16 | RW | YES | | | | ● | | ms |
| 607A | -- | VAR | target_position | INT32 | RW | YES | | | ● | | | position units |
| 607B | -- | ARRAY | position_range_limit | -- | -- | -- | | | ● | | | -- |
| | 0 | | number_of_entries | UINT8 | RW | NO | | | ● | | | -- |
| | 1 | | min_position_range_limit | INT32 | RW | NO | | | ● | | | position units |
| | 2 | | max_position_range_limit | INT32 | RW | NO | | | ● | | | position units |
| 607C | -- | VAR | home_offset | INT32 | RW | YES | | | ● | | ● | position units |
| 6081 | -- | VAR | profile_velocity | UINT32 | RW | YES | | | ● | ● | | speed units |
| 6082 | -- | VAR | end_velocity | UINT32 | RW | YES | | | ● | | | speed units |
| 6083 | -- | VAR | profile_acceleration | UINT32 | RW | YES | | | ● | ● | | acceleration |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | units |
| 6084 | -- | VAR | profile_deceleration | UINT32 | RW | YES | | | ● | ● | | acceleration units |
| 6085 | -- | VAR | quick_stop_deceleration | UINT32 | RW | YES | | | ● | ● | | acceleration units |
| 6086 | -- | VAR | motion_profile_type | INT16 | RW | YES | | | ● | ● | | -- |
| 6093 | -- | ARRAY | position_factor | -- | -- | -- | | | ● | | ● | -- |
| | 0 | | number_of_entries | UINT32 | RW | NO | | | ● | | ● | -- |
| | 1 | | numerator | UINT32 | RW | NO | | | ● | | ● | -- |
| | 2 | | divisor | UINT32 | RW | NO | | | ● | | ● | -- |
| 6094 | -- | ARRAY | velocity_encoder_factor | -- | -- | -- | ● | | | | | -- |
| | 0 | | number_of_entries | UINT32 | RW | NO | ● | | | | | -- |
| | 1 | | numerator | UINT32 | RW | NO | ● | | | | | -- |
| | 2 | | divisor | UINT32 | RW | NO | ● | | | | | |
| 6097 | -- | ARRAY | acceleration_factor | -- | -- | -- | ● | | | | | -- |
| | 0 | | number_of_entries | UINT32 | RW | NO | ● | | | | | -- |
| | 1 | | numerator | UINT32 | RW | NO | ● | | | | | -- |
| | 2 | | divisor | UINT32 | RW | NO | ● | | | | | -- |
| 6098 | -- | VAR | homing_method | INT8 | RW | YES | | | | | ● | |
| 6099 | -- | ARRAY | homing_speeds | -- | -- | -- | | | | | ● | speed units |
| | 0 | | number_of_entries | UINT8 | RW | YES | | | | | ● | |
| | 1 | | speed_during_search_for_switch | UINT32 | RW | YES | | | | | ● | speed units |
| | 2 | | speed_during_search_for_zero | UINT32 | RW | YES | | | | | ● | speed units |
| 609A | -- | VAR | homing_acceleration | UINT32 | RW | YES | | | | | ● | acceleration units |
| 60A3 | 00 | VAR | profile_jerk_use | UINT8 | RO | NO | | | ● | | | -- |
| 60A4 | -- | ARRAY | profile_jerk | -- | -- | -- | | | ● | | | -- |
| | 00 | | number of entries | UINT8 | RO | NO | | | ● | | | -- |
| | 01 | | profile_jerk1 | UINT32 | RW | NO | | | ● | | | jerk unit |
| 60C0 | -- | VAR | Interpolation sub mode select | INT16 | RW | NO | | ● | | | | |
| 60C1 | -- | ARRAY | Interpolation data record | -- | -- | -- | | ● | | | | |
| | 0 | | number_of_entries | UINT8 | -- | NO | | ● | | | | |
| | 1 | | the first parameter of ip function fip(x1, .. xN) | see 60C0h | RW | YES | | ● | | | | position units |
| | 2 | | the second parameter of ip function fip(x1, .. xN) | | RW | YES | | ● | | | | position units |
| 60C2 | -- | RECORD | Interpolation time period | -- | -- | -- | | ● | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | | ● | | | | |
| | 1 | | ip time units | UINT8 | RW | NO | | ● | | | | |
| | 2 | | ip time index | UINT8 | RW | NO | | ● | | | | |
| 60FA | -- | VAR | control_effort | INT32 | RO | YES | | | ● | | | |
| 60FC | -- | VAR | position_demand_value* | INT32 | RO | YES | | | ● | | | inc |
| 60FD | -- | VAR | digital inputs | UINT32 | RO | YES | | | | | | |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 60FE | -- | RECORD | digital outputs | -- | -- | -- | | | | | | |
| | 0 | | number_of_entries | UINT8 | RO | NO | | | | | | |
| | 1 | | pysical _outputs | UINT32 | | | | | | | | |
| | 2 | | bit_mask | UINT32 | | | | | | | | |
| 60FF | -- | VAR | target_velocity | UINT32 | RW | YES | | | | ● | | speed units |


**ESTUN AUTOMATION TECHNOLOGY CO., LTD**

Address: 16 Shuige Road, Jiangning Development Zone
Nanjing 211106, P.R.China
Tel: +86-25-58328505/8507
Fax: +86-25-58328504
Web: www.estun.cn
E-mail: export@estun.com

www.estun.cn

ESTUN