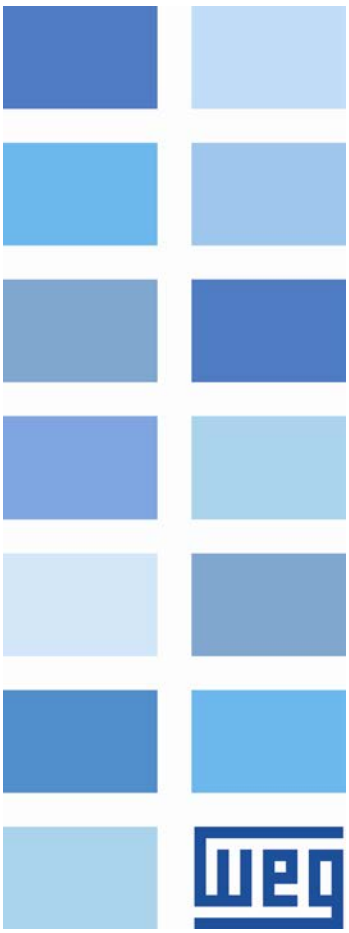
CANopen

SCA06

User's Manual



CANopen User's Manual

Series: SCA06

Language: English

Document Number: 10001615296 / 02

Publication Date: 11/2017

CONTENTS

ABOUT THE MANUAL

This manual provides the necessary information for the operation of the SCA06 frequency inverter using the CANopen protocol. This manual must be used together with the SCA06 user manual.

ABBREVIATIONS AND DEFINITIONS

CAN	Controller Area Network
CiA	CAN in Automation
COB	Communication Object
COB-ID	Communication Object Identifier
SDO	Service Data Object
PDO	Process Data Object
RPDO	Receive PDO
TPDO	Transmit PDO
NMT	Network Management Object
ro	Read only
rw	Read/write

NUMERICAL REPRESENTATION

Decimal numbers are represented by means of digits without suffix. Hexadecimal numbers are represented with the letter 'h' after the number.

DOCUMENTS

The CANopen protocol for the SCA06 was developed based on the following specifications and documents:

Document	Version	Source
CAN Specification	2.0	CiA
CiA DS 301 CANopen Application Layer and Communication Profile	4.02	CiA
CiA DRP 303-1 Cabling and Connector Pin Assignment	1.1.1	CiA
CiA DSP 306 Electronic Data Sheet Specification for CANopen	1.1	CiA
CiA DSP 402 Device Profile Drives and Motion Control	2.0	CiA

In order to obtain this documentation, the organization that maintains, publishes and updates the information regarding the CANopen network, CiA, must be consulted.

1 INTRODUCTION TO THE CANOPEN COMMUNICATION

In order to operate the equipment in a CANopen network, it is necessary to know the manner this communication is performed. Therefore, this section brings a general description of the CANopen protocol operation, containing the functions used by the SCA06. Refer to the protocol specification for a detailed description.

1.1 CAN

CANopen is a network based on CAN, i.e., it uses CAN telegrams for exchanging data in the network.

The CAN protocol is a serial communication protocol that describes the services of layer 2 of the ISO/OSI model (data link layer)[1]. This layer defines the different types of telegrams (frames), the error detection method, the validation and arbitration of messages.

1.1.1 Data Frame

CAN network data is transmitted by means of a data frame. This frame type is composed mainly by an 11 bit[2] identifier (arbitration field), and by a data field that may contain up to 8 data bytes.

Identifier	8 data bytes							
11 bits	byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7

1.1.2 Remote Frame

Besides the data frame, there is also the remote frame (RTR frame). This type of frame does not have a data field, but only the identifier. It works as a request, so that another network device transmits the desired data frame.

1.1.3 Access to the Network

Any device in a CAN network can make an attempt to transmit a frame to the network in a certain moment. If two devices try to access the network simultaneously, the one that sends the message with the highest priority will be able to transmit. The message priority is defined by the CAN frame identifier, the smaller the value of this identifier, the higher the message priority. The telegram with the identifier 0 (zero) is the one with the highest priority.

1.1.4 Error Control

The CAN specification defines several error control mechanisms, which makes the network very reliable and with a very low undetected transmission error rate. Every network device must be able to identify the occurrence of these errors, and to inform the other elements that an error was detected.

A CAN network device has internal counters that are incremented every time a transmission or reception error is detected, and are decremented when a telegram is successfully transmitted or received. If a considerable amount of errors occurs, the device can be led to the following states:

- *Error Active:* the internal error counters are at a low level and the device operates normally in the CAN network. You can send and receive telegrams and act in the CAN network if it detects any error in the transmission of telegrams.
- *Warning:* when the counter exceeds a defined limit, the device enters the *warning* state, meaning the occurrence of a high error rate.
- *Error Passive:* when this value exceeds a higher limit, the device enters the *error passive* state, and it stops acting in the network when detecting that another device sent a telegram with an error.
- *Bus Off:* finally, we have the *bus off* state, in which the device will not send or receive telegrams any more. The device operates as if disconnected from the network.

[1] In the CAN protocol specification, the ISO11898 standard is referenced as the definition of the layer 1 of this model (physical layer).
[2] The CAN 2.0 specification defines two data frame types, standard (11 bit) and extended (29 bit). For this implementation, only the standard frames are accepted.

1.1.5 CAN and CANopen

Only the definition of how to detect errors, create and transmit a frame, are not enough to define a meaning for the data transmitted via the network. It is necessary to have a specification that indicates how the identifier and the data must be assembled and how the information must be exchanged. Thus, the network elements can interpret the transmitted data correctly. In that sense, the CANopen specification defines exactly how to exchange data among the devices and how every one must interpret these data.

There are several other protocols based on CAN, as DeviceNet, CANopen, J1939, etc., which use CAN frames for the communication. However, those protocols cannot be used together in the same network.

1.2 NETWORK CHARACTERISTICS

Because of using a CAN bus as telegram transmission means, all the CANopen network devices have the same right to access the network, where the identifier priority is responsible for solving conflict problems when simultaneous access occurs. This brings the benefit of making direct communication between slaves of the network possible, besides the fact that data can be made available in a more optimized manner without the need of a master that controls all the communication performing cyclic access to all the network devices for data updating.

Another important characteristic is the use of the producer/consumer model for data transmission. This means that a message that transits in the network does not have a fixed network address as a destination. This message has an identifier that indicates what data it is transporting. Any element of the network that needs to use that information for its operation logic will be able to consume it, therefore, one message can be used by several network elements at the same time.

1.3 PHYSICAL LAYER

The physical medium for signal transmission in a CANopen network is specified by the ISO 11898 standard. It defines as transmission bus a pair of twisted wires with differential electrical signal.

1.4 ADDRESS IN THE CANOPEN NETWORK

Every CANopen network must have a master responsible for network management services, and it can also have a set of up to 127 slaves. Each network device can also be called node. Each slave is identified in a CANopen network by its address or Node-ID, which must be unique for each slave and may range from 1 to 127.

The address of servo drive SCA06 is programmed by the parameter P0701.

1.5 ACCESS TO THE DATA

Each slave of the CANopen network has a list called object dictionary that contains all the data accessible via network. Each object of this list is identified with an index, which is used during the equipment configuration as well as during message exchanges. This index is used to identify the object being transmitted.

1.6 DATA TRANSMISSION

The transmission of numerical data via CANopen telegrams is done using a hexadecimal representation of the number, and sending the least significant data byte first.

E.g: The transmission of a 32 bit integer with sign (12345678h = 305419896 decimal), plus a 16 bit integer with sign (FF00h = -256 decimal), in a CAN frame.

Identifier	6 data bytes					
11 bits	32 bit integer				16 bit integer	
	byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
	78h	56h	34h	12h	00h	FFh

1.7 COMMUNICATION OBJECTS - COB

There is a specific set of objects that are responsible for the communication among the network devices. Those objects are divided according to the type of data and the way they are sent or received by a device. The SCA06 supports the following communication objects (COB):

Table 1.1: *Types of Communication Objects (COB)*

Type of object	Description
Service Data Object (SDO)	SDO are objects responsible for the direct access to the object dictionary of a device. By means of messages using SDO, it is possible to indicate explicitly (by the object index) what data is being handled. There are two SDO types: Client SDO, responsible for doing a read or write request to a network device, and the Server SDO, responsible for taking care of that request. Since SDO are usually used for the configuration of a network node, they have less priority than other types of message.
Process Data Object (PDO)	PDO are used for accessing equipment data without the need of indicating explicitly which dictionary object is being accessed. Therefore, it is necessary to configure previously which data the PDO will be transmitting (data mapping). There are also two types of PDO: Receive PDO and Transmit PDO. They are usually utilized for transmission and reception of data used in the device operation, and for that reason they have higher priority than the SDO.
Emergency Object (EMCY)	This object is responsible for sending messages to indicate the occurrence of errors in the device. When an error occurs in a specific device (EMCY producer), it can send a message to the network. In the case that any network device be monitoring that message (EMCY consumer), it can be programmed so that an action be taken (disabling the other devices, error reset, etc.).
Synchronization Object (SYNC)	In the CANopen network, it is possible to program a device (SYNC producer) to send periodically a synchronization message for all the network devices. Those devices (SYNC consumers) will then be able, for instance, to send a certain datum that needs to be made available periodically.
Network Management (NMT)	Every CANopen network needs a master that controls the other devices (slaves) in the network. This master will be responsible for a set of services that control the slave communications and their state in the CANopen network. The slaves are responsible for receiving the commands sent by the master and for executing the requested actions. The protocol describes two types of service that the master can use: device control service, with which the master controls the state of each network slave, and error control service (Node Guarding), with which the slave sends periodic messages to the master informing that the connection is active.

All the communication of the inverter with the network is performed using those objects, and the data that can be accessed are the existent in the device object dictionary.

1.8 COB-ID

A telegram of the CANopen network is always transmitted by a communication object (COB). Every COB has an identifier that indicates the type of data that is being transported. This identifier, called COB-ID has an 11 bit size, and it is transmitted in the identifier field of a CAN telegram. It can be subdivided in two parts:

Function Code				Address						
bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- Function Code: indicates the type of object that is being transmitted.
- Node Address: indicates with which network device the telegram is linked.

A table with the standard values for the different communication objects available in the SCA06 is presented next. Notice that the standard value of the object depends on the slave address, with the exception of the COB-ID for NMT and SYNC, which are common for all the network elements. Those values can also be changed during the device configuration stage.

Table 1.2: *COB-ID for the different objects*

COB	Function code (bits 10 – 7)	Resultant COB-ID (function + address)
NMT	0000	0
SYNC	0001	128 (80h)
EMCY	0001	129 – 255 (81h – FFh)
PDO1 (tx)	0011	385 – 511 (181h – 1FFh)
PDO1 (rx)	0100	513 – 639 (201h – 27Fh)
PDO2 (tx)	0101	641 – 767 (281h – 2FFh)
PDO2 (rx)	0110	769 – 895 (301h – 37Fh)
PDO3 (tx)	0111	897 – 1023 (381h – 3FFh)
PDO3 (rx)	1000	1025 – 1151 (401h – 47Fh)
PDO4 (tx)	1001	1153 – 1279 (481h – 4FFh)
PDO4 (rx)	1010	1281 – 1407 (501h – 57Fh)
SDO (tx)	1011	1409 – 1535 (581h – 5FFh)
SDO (rx)	1100	1537 – 1663 (601h – 67Fh)
Node Guarding/Heartbeat	1110	1793 – 1919 (701h – 77Fh)

1.9 EDS FILE

Each device in a CANopen network has an EDS configuration file that contains information about the operation of the device in the CANopen network, as well as the description of all the communication objects available. In general, this file is used by a master or by the configuration software for programming of devices present in the CANopen Network.

The EDS configuration file for the SCA06 is supplied together with the product, and it can also be obtained from the website http://www.weg.net. It is necessary to observe the inverter software version, in order to use an EDS file that be compatible with that version.

2 CANOPEN COMMUNICATION INTERFACE

The standard SCA06 servo drive features a CAN interface. It can be used for communication in CANopen protocol as a network master or slave. The characteristics of this interface are described below.

2.1 CHARACTERISTICS OF THE CAN INTERFACE

Figure 2.1: *Detail of the CAN connector in the lower part of the product*

- Interface galvanically insulated and with differential signal, providing more robustness against electromagnetic interference.
- External power supply of 24 V.
- It allows the connection of up to 64 devices to the same segment. More devices can be connected by using repeaters[3].
- Maximum bus length of 1000 meters.

2.2 PIN ASSIGNMENT OF THE CONNECTOR

The CAN interface has a 5-way plug-in connector (x4) with the following pin assignment:

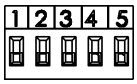


Table 2.1: *Pin assignment of X4 connector for CAN interface*

Pin	Name	Function
1	V-	Negative pole of the power supply
2	CAN_L	Communication signal CAN_L
3	Shield	Cable shield
4	CAN_H	Communication signal CAN_H
5	V+	Positive pole of the power supply

2.3 POWER SUPPLY

The CAN interfaces require an external power supply between pins 1 and 5 of the network connector. The data for individual consumption and input voltage are shown in the following table.

Table 2.2: *Characteristics of the supply for the CAN interface*

Power Supply (VDC)		
Minimum	Maximum	Recommended
11	30	24
Current (mA)		
Typical		Maximum
30		50

[3] The maximum number of devices that can be connected to the network also depends on the protocol used.

2.4 INDICATIONS

The alarm, fault and status indications of the CAN open communication for the SCA06 servo drive are made trough the HMI and parameters of the product.

3 CANOPEN NETWORK INSTALLATION

The CANopen network, such as several industrial communication networks, for being many times applied in aggressive environments with high exposure to electromagnetic interference, requires that certain precautions be taken in order to guarantee a low communication error rate during its operation. Recommendations to perform the connection of the product in this network are presented next.

3.1 BAUD RATE

Equipments with CANopen interface generally allow the configuration of the desired baud rate, ranging from 10Kbit/s to 1Mbit/s. The *baud rate* that can be used by equipment depends on the length of the cable used in the installation. The next table shows the baud rates and the maximum cable length that can be used in the installation, according to the CiA recommendation[4].

Table 3.1: *Supported baud rates and installation size*

Baud Rate	Cable Length
1 Mbit/s	25 m
800 Kbit/s	50 m
500 Kbit/s	100 m
250 Kbit/s	250 m
125 Kbit/s	500 m
100 Kbit/s	600 m
50 Kbit/s	1000 m
20 Kbit/s	1000 m
10 Kbit/s	1000 m

All network equipment must be programmed to use the same communication baud rate. At the SCA06 servo drive the baud rate configuration is done through the parameter .

3.2 ADDRESS IN THE CANOPEN NETWORK

Each CANopen network device must have an address or Node ID, and may range from 1 to 127. This address must be unique for each equipment. For SCA06 servo drive the address configuration is done through the parameter .

3.3 TERMINATION RESISTOR

The CAN bus line must be terminated with resistors to avoid line reflection, which can impair the signal and cause communication errors. The extremes of the CAN bus must have a termination resistor with a 121Ω / 0.25W value, connecting the CAN_H and CAN_L signals.

3.4 CABLE

The connection of CAN_L and CAN_H signals must done with shielded twisted pair cable. The following table shows the recommended characteristics for the cable.

Table 3.2: *CANopen cable characteristics*

Cable length (m)	Resistance per meter (mOhm/m)	Conductor cross section (mm^2)
0 ... 40	70	0.25 ... 0.34
40 ... 300	<60	0.34 ... 0.60
300 ... 600	<40	0.50 ... 0.60
600 ... 1000	<26	0.75 ... 0.80

It is necessary to use a twisted pair cable to provide additional 24Vdc power supply to equipments that need this signal. It is recommended to use a certified DeviceNet cable.

[4] Different products may have different maximum allowed cable length for installation.

3.5 CONNECTION IN THE NETWORK

In order to interconnect the several network nodes, it is recommended to connect the equipment directly to the main line without using derivations. During the cable installation the passage near to power cables must be avoided, because, due to electromagnetic interference, this makes the occurrence of transmission errors possible. In order to avoid problems with current circulation caused by difference of potential among ground connections, it is necessary that all the devices be connected to the same ground point.

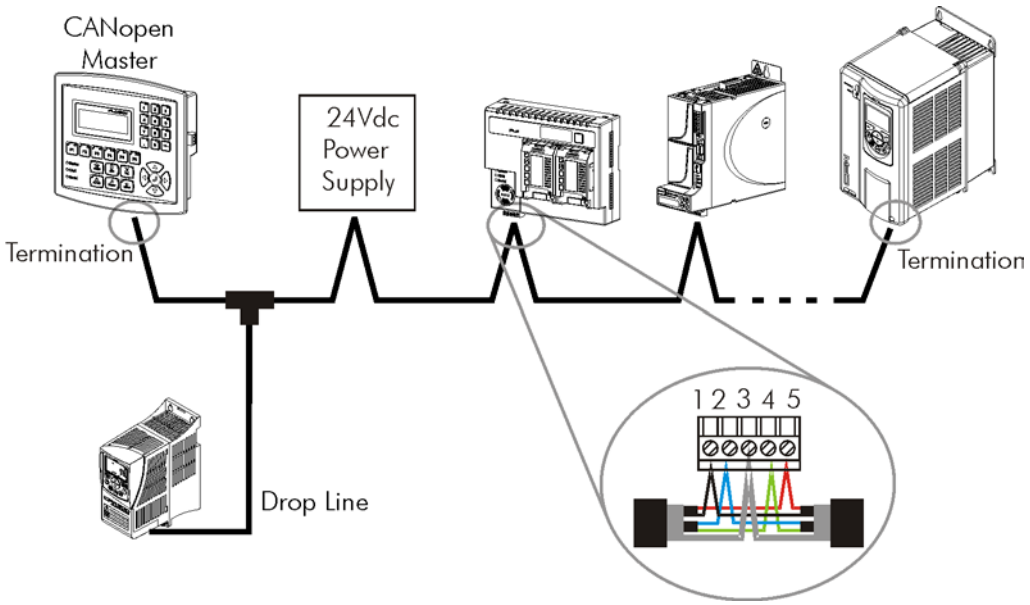


Figure 3.1: *CANopen network installation example*

To avoid voltage difference problems between the power supplies of the network devices, it is recommended that the network is fed by only one power supply and the signal is provided to all devices through the cable. If it is required more than one power supply, these should be referenced to the same point.

The maximum number of devices connected to a single segment of the network is limited to 64. Repeaters can be used for connecting a bigger number of devices.

4 PROGRAMMING

Next, only the SCA06 servo drive parameters related to the CANopen communication will be presented.

4.1 SYMBOLS FOR THE PROPERTIES DESCRIPTION

RO	Read-only parameter
CFG	Parameter that can be changed only with a stopped motor
CAN	Parameter visible on the HMI if the product has the CAN interface installed

P0070 – CAN CONTROLLER STATUS

Range: 0 = Disabled **Default:** -
1 = *Autobaud*
2 = CAN Enabled
3 = *Warning*
4 = *Error Passive*
5 = *Bus Off*
6 = No Bus Power
Properties: RO

Description:
It allows identifying if the CAN interface board is properly installed and if the communication presents errors.

Table 4.1: *Values for the parameter P0070*

Value	Description
0 = Disabled	Inactive CAN interface. It occurs when CAN protocol is not programmed at P0700.
1 = *Autobaud*	CAN controller is trying to detect baud rate of the network (only for DeviceNet communication protocol).
2 = CAN Enabled	CAN interface is active and without errors.
3 = *Warning*	CAN controller has reached the *warning* state.
4 = *Error Passive*	CAN controller has reached the *error passive* state.
5 = *Bus Off*	CAN controller has reached the *bus off* state.
6 = No Bus Power	CAN interface does not have power supply between the pins 1 and 5 of the connector.

P0071 – RECEIVED CAN TELEGRAM COUNTER

Range: 0 to 65535 **Default:** -
Properties: RO

Description:
This parameter works as a cyclic counter that is incremented every time a CAN telegram is received. It informs the operator if the device is being able to communicate with the network. This counter is reset every time the device is switched off, a reset is performed or the parameter maximum limit is reached.

P0072 – TRANSMITTED CAN TELEGRAM COUNTER

Range: 0 to 65535 **Default:** -
Properties: RO

Description:
This parameter works as a cyclic counter that is incremented every time a CAN telegram is transmitted. It informs the operator if the device is being able to communicate with the network. This counter is reset every time the device is switched off, a reset is performed or the parameter maximum limit is reached.

P0073 – BUS OFF ERROR COUNTER

Range:	0 to 65535	**Default:** -
Properties:	RO	

Description:
It is a cyclic counter that indicates the number of times the device entered the bus off state in the CAN network. This counter is reset every time the device is switched off, a reset is performed or the parameter maximum limit is reached.

P0074 – LOST CAN MESSAGE COUNTER

Range:	0 to 65535	**Default:** -
Proprerties:	RO	

Description:
It is a cyclic counter that indicates the number of messages received by the CAN interface, but could not be processed by the device. In case that the number of lost messages is frequently incremented, it is recommended to reduce the baud rate used in the CAN network. This counter is reset every time the device is switched off, a reset is performed or the parameter maximum limit is reached.

P0075 – CANOPEN COMMUNICATION STATUS

Range:	0 = Disabled 1 = Reserved 2 = Communication Enabled 3 = Error Control Enabled 4 = Guarding Error 5 = Heartbeat Error	**Default:** -
Properties:	RO, CAN	

Description:
It indicates the board state regarding the CANopen network, informing if the protocol has been enabled and if the error control service is active (*Node Guarding* or *Heartbeat*).

P0076 – CANOPEN NODE STATUS

Range:	0 = Disabled 1 = Initialization 2 = Stopped 3 = Operational 4 = Preoperational	**Default:** -
Properties:	RO, CAN	

Description:
It operates as a slave of the CANopen network, and as such element it has a state machine that controls its behavior regarding the communication. This parameter indicates in which state the device is.

P0202 – MODE OF OPERATION

Value	1 = Torque Mode	**Default:** 2
Range:	2 = Speed Mode	
	3 = Reserved	
	4 = Ladder Mode	
	5 = CANopen	
	6 = Profibus DP	
Properties:	CFG	

Description:
This parameter defines the mode of operation of the SCA06 servo drive. For the equipment to be controlled by the CANopen network, it is necessary to use mode 5 = CANopen. If this mode is programmed, commands and references for the product operation will be provided via CANopen network, using the objects defined on the object dictionary.

Among the main objects used to control and monitor the equipment, we can mention:

- 6040h: ControlWord
- 6041h: StatusWord
- 6060h: Mode of operation
- 6063h: Position actual value
- 607Ah: Target position
- 60FFh: Target velocity
- 6071h: Target Torque

The detailed description of these and other objects is found in 7. For details about the modes of operation from 1 to 4, refer to the user's manual of the SCA06 servo drive.

NOTE!
- Controlling the equipment through the objects for drives is only possible for mode of operation 5, but the CANopen communication can be used in any mode of operation.
- For the SCA06 operating as slave of the Follow function, mode of operation 4 (ladder) must be programmed, and the MC_GearInPos block must be used.

P0662 – ACTION FOR COMMUNICATION ERROR

Range:	0 = Cause Alarm	**Default:** 0
	1 = Cause Fault	
	2 = Cause alarm and execute STOP	
	3 = Cause alarm and disable drive	
Properties:	CFG	

Description:
This parameter allows selecting which action must be executed by the equipment in case it is controlled via network and a communication error is detected.

Table 4.2: *Options for the parameter P0662*

Option	Description
0 = Cause Alarm	It just indicates alarm.
1 = Cause Fault	Instead of alarm, a communication error causes a fault on the equipment, and it is necessary to reset the faults so as to return to normal operation.
2 = Execute STOP	The alarm will be indicated together with the execution of the STOP command. It is necessary to reset the faults or disable the drive for the servo to exit this condition.
3 = Disable drive	The alarm will be indicated together with the execution of the disable command.

The followings events are considered communication errors:

Serial Communication (RS232/RS485):
- Alarm A00128/Fault F00028: timeout of the serial interface.

CANopen communication:
- Alarm A133/Fault F233: no power supply on the CAN interface.
- Alarm A134/Fault F234: bus off.
- Alarm A135/Fault F235: CANopen communication error (Node Guarding/Heartbeat).

P0700 – CAN PROTOCOL

Range: 0 = Disabled **Default:** 0
1 = CANopen
2 = Reserved
3 = CANespecial 1
Properties:

Description:
It allows selecting the desired protocol for the CAN interface. If this parameter is changed, the change takes effect only if the CAN interface is not powered, it is in auto-baud or after the equipment is switched off and on again.

P0701 – CAN ADDRESS

Range: 0 to 127 **Default:** 63
Properties:

Description:
It allows programming the address used for the CAN communication. It is necessary that each element of the network has an address different from the others. The valid addresses for this parameter depend on the protocol programmed in P0700:

- P0700 = 1 (CANopen) → valid addresses: 1 to 127.

If this parameter is changed, the change takes effect only if the CAN interface is not powered, auto-baud or after the equipment is switched off and on again.

P0702 – CAN BAUD RATE

Range: 0 = 1 Mbit/s / *Autobaud* **Default:** 0
1 = 800 Kbit/s / *Autobaud*
2 = 500 Kbit/s
3 = 250 Kbit/s
4 = 125 Kbit/s
5 = 100 Kbit/s / *Autobaud*
6 = 50 Kbit/s / *Autobaud*
Properties:

Description:
It allows programming the desired baud rate for the CAN interface, in bits per second. This rate must be the same for all the devices connected to the network. The supported bauld rates for the device depend on the protocol programmed in the parameter P0700:

- P0700 = 1 (CANopen): It is possible to use any rate specified in this parameter, but it does not have the automatic baud rate detection function – *autobaud*.

If this parameter is changed, the change takes effect only if the CAN interface is not powered or after the equipment is switched off and on again.

P0703 – BUS OFF RESET

Range:	0 = Manual 1 = Automatic	**Default:** 1
Properties:		

Description:
It allows programming the inverter behavior when detecting a bus off error at the CAN interface:

Table 4.3: *Options for the parameter P0703*

Option	Description
0 = Manual Reset	If *bus off* occurs, the A134/F34 alarm will be indicated on the HMI, the action programmed in parameter P0662 will be executed and the communication will be disabled. In order that the inverter communicates again through the CAN interface, it will be necessary to cycle the power of the inverter.
1= Automatic Reset	If *bus off* occurs, the communication will be reinitiated automatically and the error will be ignored. In this case the alarm will not be indicated on the HMI and the inverter will not execute the action programmed in P0662.

P0704 – FOLLOW

Value Range:	0 = Disabled 1 = Real Follow Master 2 = Virtual Follow Master 3 = Follow Slave	**Default:** 0
Properties:	CFG	

Description:
It allows enabling the Follow function via CANopen, besides defining if the equipment will be Follow master (producer) or slave (consumer).

Table 4.1: *Options for parameter P0704*

Option	Description
0 = Disabled	It does not send Follow message
1 = Real Follow Master	It sends Follow telegrams containing position and speed of the real axis.
2 = Virtual Follow Master	It sends Follow telegrams containing position and speed of the virtual axis.
3 = Follow Slave	It receives Follow telegrams. It requires the MC_GearInPos block to execute the Follow function.

Once programmed as master or slave, the SCA06 servo drive must automatically enter the mode of operation on the CANopen network so as to enable the exchange of PDOs among the network devices.

For further details about the operation of the Follow function, refer to item 4.2.

P0705 – COB ID FOLLOW

Value Range:	385 to 511	**Default:** 0
Properties:	CFG	

Description:
It defines the COB ID (Communication Object Identifier) of the Follow PDO. The adjustable range 385 (181h) to 511 (1FFh) is defined by the CANopen specification as standard range for the TPDO1. Both the master and slaves must use the same COB ID.

The function of the TPDO1 (master) and of the RPDO1 (slaves) is dedicated to this function and, therefore, these PDOs must not be configured for communication of other data.

For details about the operation of the Follow function, refer to item 4.2.

P0706 – FOLLOW PERIOD

Value Range:	0.2 to 5.0 ms	**Default:** 1,0
Properties:	CFG	

Description:
It allows programming the transmission period of the Follow telegram by the network master. It is not used by the slaves.

The shorter the period, the faster the references are transmitted and the more accurate the synchronism. However, the bus occupation time will also be longer, which can hinder the communication in case there are other data to be communicated by the CANopen network. This period must also be programmed considering the baud rate. At 1 Mbit/s, it takes a Follow telegram about 100 us to be transmitted. As the baud rate gets slower, the transmission time increases proportionally.

If the Follow function is used on the network only, you can program the shortest possible period, since there will not be other transmitted telegrams on the network. But if the CANopen master function is used in parallel, it is important that there is available time to transmit the other CANopen telegrams. As a recommendation, when the CANopen master is used, the Follow telegrams must take about 10 to 20% of the bus time.

For further details about the operation of the Follow function, refer to item 4.2.

4.2 FOLLOW FUNCTION VIA CANOPEN

The Follow function allows the position synchronism between two or more servomotors. The synchronism is established by sending the telegrams of the PDO type, where the Follow master sends position and speed values of the motor, which will be used as reference by one or more Follow slaves.

The Follow function can be programmed by two different sources: by parameters or by the WSCAN software.

4.2.1 Follow Programmed by Parameters

If you do not wish to use the CANopen master function available for the SCA06 servo drive, you can program the Follow function only by using parameters. In this case, parameters P0704, P0705 and P0706 are used, and it is necessary to program/configure the following elements:

- Network: In order to use the Follow function, first it is necessary to configure the CANopen interface, defining the protocol, address and baud rate, as well as to perform the necessary installation for the communication – cables, power supply, termination resistors, etc.
- Master: For the master (or producer), it is necessary to enable the Follow function, select the reference shaft, and program the COB ID and the transmission period.
- Slaves: On slaves (or consumers), you must enable the Follow function in the slave mode and program the COB ID equal to that programmed for the master. For the slaves to receive and use these speed and position values, it is also necessary that they be programmed for the Ladder mode (P0204 = 4), and the MC_GearInPos block must be used. More details about the configuration of the MC_GearInPos block in the Help of the WLP software.

4.2.2 Follow Programmed by the WSCAN Software

Another option for programming the Follow function is trough the WSCAN software. In this case, the parameters P0704, P0705 and P0706 must not be programmed, and the function is enabled through the SCA06 servo drive configuration window in the WSCAN software.

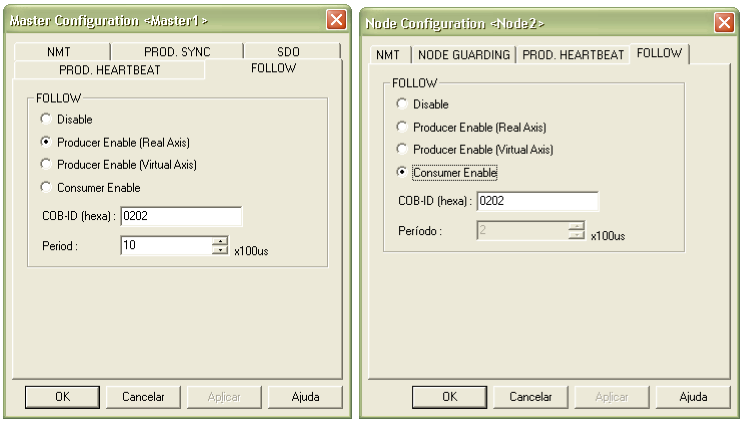


Figure 4.1: *Configuration windows for Follow master and slave*

Once programmed, during the initialization of the slaves with the Follow function, the CANopen network master will program the function by using internal objects of the device. The programming is done online, and it is not necessary to restart the equipment for the new values to be used.

Along with the programming of the Follow function, you can also program other services for the CANopen network, such as PDOs, SDOs, Node Guarding, etc., always bearing in mind that the CAN bus occupation time by the Follow telegrams must allow for the use of these other services.

5 OBJECT DICTIONARY

The object dictionary is a list containing several equipment data which can be accessed via CANopen network. An object of this list is identified by means of a 16-bit index, and it is based in that list that all the data exchange between devices is performed.

The CiA DS 301 document defines a set of minimum objects that every CANopen network slave must have. The objects available in that list are grouped according to the type of function they execute. The objects are arranged in the dictionary in the following manner:

Table 5.1: *Object dictionary groupings*

Index	Objects	Description
0001h – 025Fh	Data type definition	Used as reference for the data type supported by the system.
1000h – 1FFFh	Communication objects	They are objects common to all the CANopen devices. They contain general information about the equipment and also data for the communication configuration.
2000h – 5FFFh	Manufacturer specific objects	In this range, each CANopen equipment manufacturer is free to define which data those objects will represent.
6000h – 9FFFh	Standardized device objects	This range is reserved to objects that describe the behavior of similar equipment, regardless of the manufacturer.

The other indexes that are not referred in this list are reserved for future use.

5.1 DICTIONARY STRUCTURE

The general structure of the dictionary has the following format:

Index	Object	Name	Type	Access

- Index: indicates directly the object index in the dictionary.
- Object: describes which information the index stores (simple variable, array, record, etc.).
- Name: contains the name of the object in order to facilitate its identification.
- Type: indicates directly the stored data type. For simple variables, this type may be an integer, a float, etc. For arrays, it indicates the type of data contained in the array. For records, it indicates the record format according to the types described in the first part of the object dictionary (indexes 0001h – 0360h).
- Access: informs if the object in question is accessible only for reading (ro), for reading and writing (rw), or if it is a constant (const).

For objects of the array or record type, a sub-index that is not described in the dictionary structure is also necessary.

5.2 DATA TYPE

The first part of the object dictionary (index 0001h – 025Fh) describes the data types that can be accessed at a CANopen network device. They can be basic types, as integers and floats, or compound types formed by a set of entries, as records and arrays.

5.3 COMMUNICATION PROFILE – COMMUNICATION OBJECTS

The indexes from 1000h to 1FFFh in the object dictionary correspond to the part responsible for the CANopen network communication configuration. Those objects are common to all the devices, however only a few are obligatory. A list with the objects of this range that are supported by the servo drive SCA06, working in slave mode.is presented next.

Table 5.2: *Object list – Communication Profile*

Índice	Objeto	Nome	Tipo	Acesso
1000h	VAR	device type	UNSIGNED32	ro
1001h	VAR	error register	UNSIGNED8	ro
1005h	VAR	COB-ID SYNC	UNSIGNED32	rw
100Ch	VAR	guard time	UNSIGNED16	rw
100Dh	VAR	life time factor	UNSIGNED8	rw
1016h	ARRAY	Consumer heartbeat time	UNSIGNED32	rw
1017h	VAR	Producer heartbeat time	UNSIGNED16	rw
1018h	RECORD	Identity Object	Identity	ro
Server SDO Parameter				
1200h	RECORD	1st Server SDO parameter	SDO Parameter	ro
Receive PDO Communication Parameter				
1400h	RECORD	1st receive PDO Parameter	PDO CommPar	rw
1401h	RECORD	2nd receive PDO Parameter	PDO CommPar	rw
...				
1407h	RECORD	8th receive PDO Parameter	PDO CommPar	rw
Receive PDO Mapping Parameter				
1600h	RECORD	1st receive PDO mapping	PDO Mapping	rw
1601h	RECORD	2nd receive PDO mapping	PDO Mapping	rw
...				
1607h	RECORD	8th receive PDO mapping	PDO Mapping	rw
Transmit PDO Communication Parameter				
1800h	RECORD	1st transmit PDO Parameter	PDO CommPar	rw
1801h	RECORD	2nd transmit PDO Parameter	PDO CommPar	rw
...				
1807h	RECORD	8th transmit PDO Parameter	PDO CommPar	rw
Transmit PDO Mapping Parameter				
1A00h	RECORD	1st transmit PDO mapping	PDO Mapping	rw
1A01h	RECORD	2nd transmit PDO mapping	PDO Mapping	rw
...				
1A07h	RECORD	8th transmit PDO mapping	PDO Mapping	rw

These objects can only be read and written via the CANopen network, it is not available via the keypad (HMI) or other network interface. The network master, in general, is the equipment responsible for setting up the equipment before starting the operation. The EDS configuration file brings the list of all supported communication objects.

Refer to item 6 for more details on the available objects in this range of the objects dictionary.

5.4 MANUFACTURER SPECIFIC – SCA06 SPECIFIC OBJECTS

For indexes from 2000h to 5FFFh, each manufacture is free to define which objects will be present, and also the type and function of each one. In the case of the SCA06, the whole list of parameters was made available in this object range. It is possible to operate the SCA06 by means of these parameters, carrying out any function that the inverter can execute. The parameters were made available starting from the index 2000h, and by adding their number to this index their position in the dictionary is obtained. The next table illustrates how the parameters are distributed in the object dictionary.

Table 5.3: *SCA06 object list – Manufacturer Specific*

Index	Object	Name	Type	Access
2000h	VAR	P0000 – Access parameter	INTEGER16	rw
2002h	VAR	P0002 – Motor speed	INTEGER16	ro
2003h	VAR	P0003 – Motor current	INTEGER16	ro
2004h	VAR	P0004 – DC voltage	INTEGER16	ro
...	...	...	...	...
2077h	VAR	P0119 – Current Reference	INTEGER16	rw
2079h	VAR	P0121 – Speed Reference	INTEGER16	rw
...	...	...	...	...

Refer to the SCA06 manual for a complete list of the parameters and their detailed description. In order to be able to program the device operation correctly via the CANopen network, it is necessary to know its operation through the parameters.

Besides parameters, SCA06 also have the following objects:

- 0x3000h – digital inputs.
- 0x3001h – digital outputs.
- 0x3002h – Follow Position Actual Value.
- 0x3003h – Follow Velocity Actual Value.
- 0x3004h – Follow Target Position.
- 0x3005h – Follow Target Velocity.
- 0x3008h – Follow Type.
- 0x3009h – Follow Period.

5.4.1 Objeto 3000h – Digital Inputs

This object allows reading the digital inputs status from SCA06 servo drive.

Index	3000h
Name	Digital Inputs
Object	Array
Type	UNSIGNED16

Sub-index	0
Description	Number of Entries
Access	ro
PDO Mapping	No
Range	UNSIGNED16

Sub-index	1
Description	Standard Digital Inputs
Access	ro
PDO Mapping	Yes
Range	UNSIGNED16

Sub-index	2
Description	Digital Inputs Slot 1
Access	ro
PDO Mapping	Yes
Range	UNSIGNED16

Sub-index	3
Description	Digital Inputs Slot 2
Access	ro
PDO Mapping	Yes
Range	UNSIGNED16

Sub-index	4
Description	Digital Inputs Slot 3
Access	ro
PDO Mapping	Yes
Range	UNSIGNED16

5.4.2 Objeto 3001h – Digital Outputs

This object allows writing the digital output values to SCA06 servo drive.

Index	3001h
Name	Digital Outputs
Object	Array
Type	UNSIGNED8

Sub-index	0
Description	Number of Entries
Access	ro
PDO Mapping	No
Range	UNSIGNED16

Sub-index	1
Description	Standard Digital Outputs
Access	rw
PDO Mapping	Yes
Range	UNSIGNED8

Sub-index	2
Description	Digital Outputs Slot 1
Access	rw
PDO Mapping	Yes
Range	UNSIGNED8

Sub-index	3
Description	Digital Outputs Slot 2
Access	rw
PDO Mapping	Yes
Range	UNSIGNED8

Sub-index	4
Description	Digital Outputs Slot 3
Access	rw
PDO Mapping	Yes
Range	UNSIGNED8

5.4.3 Objects 3002h to 3009h – Follow

Objects 3002h to 3009h are used for Follow function.

5.5 DEVICE PROFILE – COMMON OBJECTS FOR DRIVES

The CANopen documentation also includes suggestions for standardization of certain device types. The SCA06 servo drive follows the *CiA DPS 402 – Device Profile Drives and Motion Control* description. This document describes a set of objects that must be common for drives, regardless of the manufacturer. This makes the interaction between devices with the same function easier (as for servo drives), because the data, as well as the device behavior, are made available in a standardized manner. For those objects the indexes from 6000h to 9FFFh were reserved.

Refer to the section 7 for a detailed description of which objects are available for this range of the object dictionary.

6 COMMUNICATION OBJECTS DESCRIPTION

This item describes in detail each of the communication objects available for the servo drive SCA06 working in slave mode. It is necessary to know how to operate these objects to be able to use the available functions for the inverter communication.

NOTE!
The servo drive SCA06 can operate as master or slave of the CANopen network. The objects below describe the operation of the equipment as slave of the CANopen network. For a description of the characteristics of the product operating as CANopen network master, refer to the item 8 together with the WSCAN CANopen network configuration software.

6.1 IDENTIFICATION OBJECTS

There is a set of objects in the dictionary which are used for equipment identification; however, they do not have influence on their behavior in the CANopen network.

6.1.1 Object 1000h – Device Type

This object gives a 32-bit code that describes the type of object and its functionality.

Index	1000h
Name	Device type
Object	VAR
Type	UNSIGNED32

Access	ro
PDO Mapping	No
Range	UNSIGNED32
Default value	0002.0192h

This code can be divided into two parts: 16 low-order bits describing the type of profile that the device uses, and 16 high-order bits indicating a specific function according to the specified profile.

6.1.2 Object 1001h – Error Register

This object indicates whether or not an error in the device occurred. The type of error registered for the SCA06 follows what is described in the table 6.1.

Index	1001h
Name	Error register
Object	VAR
Type	UNSIGNED8

Access	ro
PDO Mapping	Yes
Range	UNSIGNED8
Default value	0

Table 6.1: *Structure of the object Error Register*

Bit	Meaning
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication
5	Reserved (always 0)
6	Reserved (always 0)
7	Specific of the manufacturer

If the device presents any error, the equivalent bit must be activated. The first bit (generic error) must be activated with any error condition.

6.1.3 Object 1018h – Identity Object

It brings general information about the device.

Index	1018h
Name	Identity object
Object	Record
Type	Identity

Sub index	0
Description	Number of the last sub-index
Access	RO
PDO Mapping	No
Range	UNSIGNED8
Default value	4

Sub index	1
Description	Vendor ID
Access	RO
PDO Mapping	No
Range	UNSIGNED32
Default value	0000.0123h

Sub index	2
Description	Product code
Access	RO
PDO Mapping	No
Range	UNSIGNED32
Default value	0000.0700h

Sub index	3
Description	Revision number
Access	RO
PDO Mapping	No
Range	UNSIGNED32
Default value	According to the equipment firmware version

Sub index	4
Description	Serial number
Access	RO
PDO Mapping	No
Range	UNSIGNED32
Default value	Different for every SCA06

The vendor ID is the number that identifies the manufacturer at the CiA. The product code is defined by the manufacturer according to the type of product. The revision number represents the equipment firmware version. The sub-index 4 is a unique serial number for each servo drive SCA06 in CANopen network.

6.2 SERVICE DATA OBJECTS – SDOS

The SDOs are responsible for the direct access to the object dictionary of a specific device in the network. They are used for the configuration and therefore have low priority, since they do not have to be used for communicating data necessary for the device operation.

There are two types of SDOs: client and server. Basically, the communication initiates with the client (usually the master of the network) making a read (*upload*) or write (*download*) request to a server, and then this server answers the request.

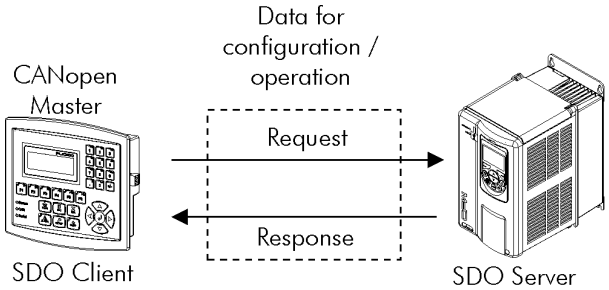


Figure 6.1: *Communication between SDO client and server*

6.2.1 Object 1200h – SDO Server

The servo drive SCA06 working in slave mode has only one SDO of the server type, which makes it possible the access to its entire object dictionary. Through it, an SDO client can configure the communication, the parameters and the drive operation. Every SDO server has an object, of the SDO_PARAMETER type, for its configuration, having the following structure:

Index	1200h
Name	Server SDO Parameter
Object	Record
Type	SDO Parameter

Sub index	0
Description	Number of the last sub-index
Access	RO
PDO Mapping	No
Range	UNSIGNED8
Default value	2

Sub index	1
Description	COB-ID Client - Server (rx)
Access	RO
PDO Mapping	No
Range	UNSIGNED32
Default value	600h + Node-ID

Sub index	2
Description	COB-ID Server - Client (tx)
Access	RO
PDO Mapping	No
Range	UNSIGNED32
Default value	580h + Node-ID

6.2.2 SDOs Operation

A telegram sent by an SDO has an 8 byte size, with the following structure:

Identifier	8 data bytes							
11 bits	Command	Index		Sub-index	Object data			
	byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7

The identifier depends on the transmission direction (rx or tx) and on the address (or Node-ID) of the destination server. For instance, a client that makes a request to a server which Node-ID is 1, must send a message with the identifier 601h. The server will receive this message and answer with a telegram which COB-ID is equal to 581h.

The command code depends on the used function type. For the transmissions from a client to a server, the following commands can be used:

Table 6.2: *Command codes for SDO client*

Command	Function	Description	Object data
22h	Download	Write object	Not defined
23h	Download	Write object	4 bytes
2Bh	Download	Write object	2 bytes
2Fh	Download	Write object	1 byte
40h	Upload	Read object	Not used
60h or 70h	Upload segment	Segmented read	Not used

When making a request, the client will indicate through its COB-ID, the address of the slave to which this request is destined. Only a slave (using its respective SDO server) will be able to answer the received telegram to the client. The answer telegram will have also the same structure of the request telegram, the commands however are different:

Table 6.3: *Command codes for SDO server*

Command	Function	Description	Object data
60h	Download	Response to write object	Not used
43h	Upload	Response to read object	4 bytes
4Bh	Upload	Response to read object	2 bytes
4Fh	Upload	Response to read object	1 byte
41h	Upload segment	Initiates segmented response for read	4 bytes
01h ... 0Dh	Upload segment	Last data segment for read	8 ... 2 bytes

For readings of up to four data bytes, a single message can be transmitted by the server; for the reading of a bigger quantity of bytes, it is necessary that the client and the server exchange multiple telegrams.

A telegram is only completed after the acknowledgement of the server to the request of the client. If any error is detected during telegram exchanges (for instance, no answer from the server), the client will be able to abort the process by means of a warning message with the command code equal to 80h.

NOTE!
When the SDO is used for writing in objects that represent the SCA06 parameters (objects starting from the index 2000h), this value is saved in the nonvolatile frequency inverter memory. Therefore, the configured values are not lost after the equipment is switched off or reset. For all the other objects these values are not saved automatically, so that it is necessary to rewrite the desired values

E.g.: A client SDO requests for a SCA06 at address 1 the reading of the object identified by the index 2000h, sub-index 0 (zero), which represents an 16-bit integer. The master telegram has the following format:

Identifier	Command	Index		Sub-index	Data			
601h	40h	00h	20h	00h	00h	00h	00h	00h

The SCA06 responds to the request indicating that the value of the referred object is equal to 999[5]:

Identifier	Command	Index		Sub-index	Data			
581h	4Bh	00h	20h	00h	E7	03h	00h	00h

6.3 PROCESS DATA OBJECTS – PDOS

The PDOs are used to send and receive data used during the device operation, which must often be transmitted in a fast and efficient manner. Therefore, they have a higher priority than the SDOs.

In the PDOs only data are transmitted in the telegram (index and sub-index are omitted), and in this way it is possible to do a more efficient transmission, with larger volume of data in a single telegram. However it is necessary to configure previously what is being transmitted by the PDO, so that even without the indication of the index and sub-index, it is possible to know the content of the telegram.

There are two types of PDOs, the receive PDO and the transmit PDO. The transmit PDOs are responsible for sending data to the network, whereas the receive PDOs remain responsible for receiving and handling these

[5] Do not forget that for any integer type of data, the byte transfer order is from the least significant to the most significant.

data. In this way it is possible to have communication among slaves of the CANopen network, it is only necessary to configure one slave to transmit information and one or more slaves to receive this information.

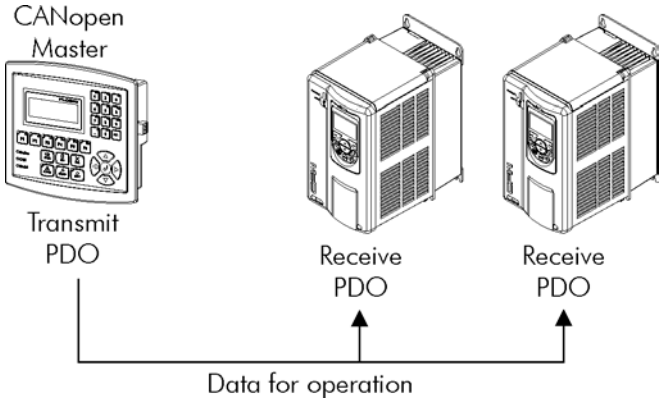


Figure 6.2: *Communication using PDOs*

NOTE!
PDOs can only be transmitted or received when the device is in the operational state. The figure 6.2 illustrates the available states for CANopen network node.

6.3.1 PDO Mapping Objects

In order to be able to be transmitted by a PDO, it is necessary that an object be mapped into this PDO content. In the description of communication objects (1000h – 1FFFh), the filed “PDO Mapping” informs this possibility. Usually only information necessary for the operation of the device can be mapped, such as enabling commands, device status, reference, etc. Information on the device configuration are not accessible through PDOs, and if it is necessary to access them one must use the SDOs.

For SCA06 specific objects (2000h – 5FFFh), the next table presents some PDO mapping objects. Read-only parameters (ro) can be used only by transmit PDOs, whereas the other parameters can be used only by receive PDOs. The SCA06 EDS file brings the list of all the objects available for the inverter, informing whether the object can be mapped or not.

Table 6.4: *Examples of PDO mapping parameters*

Index	Object	Name	Type	Access
2002h	VAR	P0002 – Motor speed	INTEGER16	ro
2003h	VAR	P0003 – Motor current	INTEGER16	ro
2004h	VAR	P0004 – DC Link Voltage (Ud)	UNSIGNED16	ro
2008h	VAR	P0008 – DI1 to DI3 status	UNSIGNED16	ro
2063h	VAR	P0099 – Enable	UNSIGNED16	rw
2077h	VAR	P0119 – Current Reference	INTEGER16	rw
2079h	VAR	P0121 – Speed Reference	INTEGER16	rw

The EDS file brings the list of all available objects informing whether the object can be mapped or not.

6.3.2 Receive PDOs

The receive PDOs, or RPDOs, are responsible for receiving data that other devices send to the CANopen network. The servo drive SCA06 working in slave mode has 8 receive PDOs, each one being able to receive up to 8 bytes. Each RPDO has two parameters for its configuration, a PDO_COMM_PARAMETER and a PDO_MAPPING, as described next.

PDO_COMM_PARAMETER

Index	1400h up to 1407h
Name	Receive PDO communication parameter
Object	Record
Type	PDO COMM PARAMETER

Sub index	0
Description	Number of the last sub-index
Access	ro
PDO Mapping	No
Range	UNSIGNED8
Default value	2

Sub index	1
Description	COB-ID used by the PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1400h: 200h + Node-ID 1401h: 300h + Node-ID 1402h: 400h + Node-ID 1403h: 500h + Node-ID 1404h – 1407h: 0

Sub index	2
Description	Transmission Type
Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	254

The sub-index 1 contains the receive PDO COB-ID. Every time a message is sent to the network, this object will read the COB-ID of that message and, if it is equal to the value of this field, the message will be received by the device. This field is formed by an UNSIGNED32 with the following structure:

Table 6.5: *COB-ID description*

Bit	Value	Description
31 (MSB)	0	PDO is enabled
	1	PDO is disabled
30	0	RTR permitted
29	0	Identifier size = 11 bits
28 – 11	0	Not used, always 0
10 – 0 (LSB)	X	11-bit COB-ID

The bit 31 allows enabling or disabling the PDO. The bits 29 and 30 must be kept in 0 (zero), they indicate respectively that the PDO accepts remote frames (RTR frames) and that it uses an 11-bit identifier. Since the SCA06 frequency inverter does not use 29-bit identifiers, the bits from 28 to 11 must be kept in 0 (zero), whereas the bits from 10 to 0 (zero) are used to configure the COB-ID for the PDO.

The sub-index 2 indicates the transmission type of this object, according to the next table.

Table 6.6: *Description of the type of transmission*

Type of transmission	PDOs transmission				
	Cyclic	Acyclic	Synchronous	Asynchronous	RTR
0		•	•		
1 – 240	•		•		
241 – 251	Reserved				
252			•		•
253				•	•
254				•	
255				•	

- Values 0 – 240: any RPDO programmed in this range presents the same performance. When detecting a message, it will receive the data; however it won't update the received values until detecting the next SYNC telegram.
- Values 252 and 253: not allowed for receive PDOs.
- Values 254 and 255: they indicated that there is no relationship with the synchronization object. When receiving a message, its values are updated immediately.

PDO_MAPPING

Index	1600h up to 1607h
Name	Receive PDO mapping
Object	Record
Type	PDO MAPPING

Sub index	0
Description	Number of mapped objects
Access	RO
PDO Mapping	No
Range	0 = disable 1 ... 4 = number of mapped objects
Default value	0

Sub index	1 up to 4
Description	1 up to 4 object mapped in the PDO
Access	Rw
PDO Mapping	No
Range	UNSIGNED32
Default value	According EDS file

This parameter indicates the mapped objects in the SCA06 receive PDOs. It is possible to map up to 4 different objects for each RPDO, provided that the total length does not exceed eight bytes. The mapping of an object is done indicating its index, sub-index[6] and size (in bits) in an UNSIGNED32, field with the following format:

UNSIGNED32		
Index (16 bits)	Sub-index (8 bits)	Size of the object (8 bits)

For instance, analyzing the receive PDO standard mapping, we have:

- Sub-index 0 = 2: the RPDO has two mapped objects.
- Sub-index 1 = 2063.0010h: the first mapped object has an index equal to 2063h, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the inverter parameter P0099, which represents the drive enabling command.
- Sub-index 2 = 2079.0010h: the second mapped object has an index equal to 2079h, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the inverter parameter P0121, which represents the speed reference.

It is possible to modify this mapping by changing the quantity or the number of mapped objects. Remembering that only 84 objects or 8 bytes can be mapped at maximum.

NOTE!

- In order to change the mapped objects in a PDO, it is first necessary to write the value 0 (zero) in the sub-index 0 (zero). In that way the values of the sub-indexes 1 to 8 can be changed. After the desired mapping has been done, one must write again in the sub-index 0 (zero) the number of objects that have been mapped, enabling again the PDO.
- In order to speed up the updating of data via PDO, the values received with these objects are not saved in the inverter non-volatile memory. Therefore, after switching off or resetting the equipment the objects modified by an RPDO get back to their default value.
- Do not forget that PDOs can only be received if the SCA06 is in the operational state.

6.3.3 Transmit PDOs

The transmit PDOs, or TPDOs, as the name says, are responsible for transmitting data for the CANopen network. The servo drive SCA06 has transmit PDOs, each one being able to transmit up to 8 data bytes. In a manner similar to RPDOs, each TPDO has two parameters for its configuration, a PDO_COMM_PARAMETER and a PDO_MAPPING, AS DESCRIBED NEXT.

PDO_COMM_PARAMETER

[6] If the object is of the VAR type and does not have sub-index, the value 0 (zero) must be indicated for the sub-index.

Index	1800h up to 1807h
Name	Transmit PDO Parameter
Object	Record
Type	PDO COMM PARAMETER

Sub index	0
Description	Number of the last sub-index
Access	ro
PDO Mapping	No
Range	UNSIGNED8
Default value	5

Sub index	1
Description	COB-ID used by the PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1800h: 180h + Node-ID 1801h: 280h + Node-ID 1802h: 380h + Node-ID 1803h: 480h + Node-ID 1804h – 1807h: 0

Sub index	2
Description	Transmission Type
Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	254

Sub index	3
Description	Time between transmissions
Access	rw
PDO Mapping	No
Range	UNSIGNED16
Default value	-

Sub index	4
Description	Reserved
Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	-

Sub index	5
Description	Event timer
Access	rw
PDO Mapping	No
Range	0 = disable UNSIGNED16
Default value	0

The sub-index 1 contains the transmit PDO COB-ID. Every time this PDO sends a message to the network, the identifier of that message will be this COB-ID. The structure of this field is described in table 6.5.

The sub-index 2 indicates the transmission type of this object, which follows the table 6.6 description. Its working is however different for transmit PDOs:

- Value 0: indicates that the transmission must occur immediately after the reception of a SYNC telegram, but not periodically.
- Values 1 – 240: the PDO must be transmitted at each detected SYNC telegram (or multiple occurrences of SYNC, according to the number chosen between 1 and 240).
- Value 252: indicates that the message content must be updated (but not sent) after the reception of a SYNC telegram. The transmission of the message must be done after the reception of a remote frame (RTR frame).
- Value 253: the PDO must update and send a message as soon as it receives a remote frame.
- Values 254: The object must be transmitted according to the timer programmed in sub-index 5.

- Values 255: the object is transmitted automatically when the value of any of the objects mapped in this PDO is changed. It works by changing the state (*Change of State*). This type does also allow that the PDO be transmitted according to the timer programmed in sub-index 5.

In the sub-index 3 it is possible to program a minimum time (in multiples of 100µs) that must elapse after the a telegram has been sent, so that a new one can be sent by this PDO. The value 0 (zero) disables this function.

The sub-index 5 contains a value to enable a timer for the automatic sending of a PDO. Therefore, whenever a PDO is configured as the asynchronous type, it is possible to program the value of this timer (in multiples of 1ms), so that the PDO is transmitted periodically in the programmed time.

NOTE!
- The value of this timer must be programmed according to the used transmission rate. Very short times (close to the transmission time of the telegram) are able to monopolize the bus, causing indefinite retransmission of the PDO, and avoiding that other less priority objects transmit their data.
- The minimum time allowed for this Function in the servo drive SCA06 is 1ms.
- It is important to observe the time between transmissions programmed in the sub-index 3, especially when the PDO is programmed with the value 255 in the sub-index 2 (*Change of State*).

PDO_MAPPING

Index	1A00h up to 1A07h
Name	Transmit PDO mapping
Object	Record
Type	PDO MAPPING

Sub index	0
Description	Number of the last sub-index
Access	ro
PDO Mapping	No
Range	0 = disable 1 ... 4 = number of mapped objects
Default value	0

Sub index	1 up to 4
Description	1 up to 4 object mapped in the PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	0

The PDO MAPPING for the transmission works in similar way than for the reception, however in this case the data to be transmitted by the PDO are defined. Each mapped object must be put in the list according to the description showed next:

UNSIGNED32		
Index (16 bits)	Sub-index (8 bits)	Size of the object (8 bits)

For instance, analyzing the standard mapping of the fourth transmit PDO, we have:

- *Sub- index 0 = 2*: This PDO has two mapped objects.
- *Sub- index 1 = 2002.0010h*: the first mapped object has an index equal to 2002h, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P0002 that is motor speed.
- *Sub-índice 2 = 2023.0010h*: the second mapped object has an index equal to 2023h, sub- index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P0035 that is present fault.

Therefore, every time this PDO transmits its data, it elaborates its telegram containing four data bytes, with the values of the parameters P0680 and P0681. It is possible to modify this mapping by changing the quantity or the number of mapped objects. Remember that a maximum of 4 objects or 8 bytes can be mapped.

NOTE!
In order to change the mapped objects in a PDO, it is first necessary to write the value 0 (zero) in the sub-index 0 (zero). In that way the values of the sub-indexes 1 to 4 can be changed. After the desired mapping has been done, one must write again in the sub-index 0 (zero) the number of objects that have been mapped, enabling again the PDO.

6.4 SYNCHRONIZATION OBJECT – SYNC

This object is transmitted with the purpose of allowing the synchronization of events among the CANopen network devices. It is transmitted by a SYNC producer, and the devices that detect its transmission are named SYNC consumers

The servo drive SCA06 working in slave mode has the function of a SYNC consumer and, therefore, it can program its PDOs to be synchronous. As described in table 6.6, synchronous PDOs are those related to the synchronization object, thus they can be programmed to be transmitted or updated based in this object.

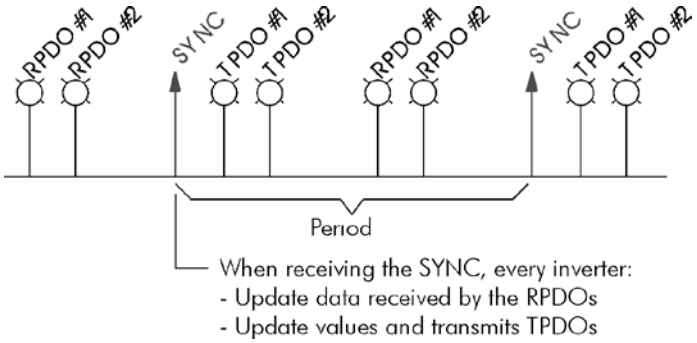


Figure 6.3: *SYNC*

The SYNC message transmitted by the producer does not have any data in its data field, because its purpose is to provide a time base for the other objects. There is an object in the SCA06 for the configuration of the COB-ID of the SYNC consumer.

Index	1015h
Name	COB-ID SYNC
Object	VAR
Type	UNSIGNED32

Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	80h

NOTE!
The period of the SYNC telegrams must be programmed in the producer according to the transmission rate and the number of synchronous PDOs to be transmitted. There must be enough time for the transmission of these objects, and it is also recommended that there is a tolerance to make it possible the transmission of asynchronous messages, such as EMCY, asynchronous PDOs and SDOs.

6.5 NETWORK MANAGEMENT – NMT

The network management object is responsible for a series of services that control the communication of the device in a CANopen network. For the SCA06 the services of node control and error control are available (using *Node Guarding* or *Heartbeat*).

6.5.1 Slave State Control

With respect to the communication, a CANopen network device can be described by the following state machine:

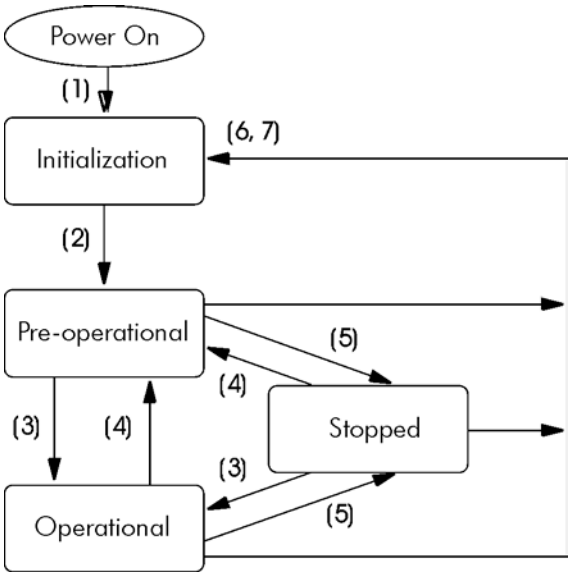


Figure 6.4: *CANopen node state diagram*

Table 6.7: *Transitions Description*

Transition	Description
1	The device is switched on and initiates the initialization (automatic).
2	Initialization concluded, it goes to the preoperational state (automatic).
3	It receives the Start Node command for entering the operational state.
4	It receives the Enter Pre-Operational command, and goes to the preoperational state.
5	It receives the Stop Node command for entering the stopped state.
6	It receives the Reset Node command, when it executes the device complete reset.
7	It receives the Reset Communication command, when it reinitializes the object values and the CANopen device communication.

During the initialization the Node-ID is defined, the objects are created and the interface with the CAN network is configured. Communication with the device is not possible during this stage, which is concluded automatically. At the end of this stage the slave sends to the network a telegram of the Boot-up Object, used only to indicate that the initialization has been concluded and that the slave has entered the preoperational state. This telegram has the identifier 700h + Node-ID, and only one data byte with value equal to 0 (zero).

In the preoperational state it is already possible to communicate with the slave, but its PDOs are not yet available for operation. In the operational state all the objects are available, whereas in the stopped state only the NMT object can receive or transmit telegrams to the network. The next table shows the objects available for each state.

Table 6.8: *Objects accessible in each state*

	Initialization	Preoperational	Operational	Stopped
PDO			•	
SDO		•	•	
SYNC		•	•	
EMCY		•	•	
Boot-up	•			
NMT		•	•	•

This state machine is controlled by the network master, which sends to each slave the commands so that the desired state change be executed. These telegrams do not have confirmation, what means that the slave does only receive the telegram without returning an answer to the master. The received telegrams have the following structure:

Identifier	byte 1	byte 2
00h	Command code	Destination Node-ID

Table 6.9: *Commands for the state transition*

Command code	Destination Node-ID
1 = START node (transition 3) 2 = STOP node (transition 4) 128 = Enter pre-operational (transition 5) 129 = Reset node (transition 6) 130 = Reset communication (transition 7)	0 = All the slaves 1 ... 127 = Specific slave

The transitions indicated in the command code correspond to the state transitions executed by the node after receiving the command (according to the Figure 6.4). The *Reset node* command makes the SCA06 execute a complete reset of the device, while the *Reset communication* command causes the device to reinitialize only the objects pertinent to the CANopen communication.

6.5.2 Error Control – Node Guarding

This service is used to make it possible the monitoring of the communication with the CANopen network, both by the master and the slave as well. In this type of service the master sends periodical telegrams to the slave, which responds to the received telegram. If some error that interrupts the communication occurs, it will be possible to identify this error, because the master as well as the slave will be notified by the *Timeout* in the execution of this service. The error events are called *Node Guarding* for the master and *Life Guarding* for the slave.

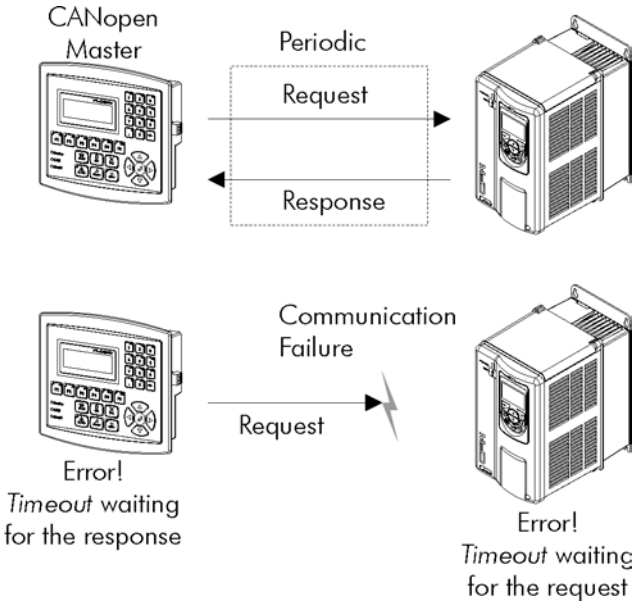


Figure 6.5: *Error control service – Node Guarding*

There are two objects of the dictionary for the configuration of the error detection times for the *Node Guarding* service:

Index	100Ch
Name	Guard Time
Object	VAR
Type	UNSIGNED16

Access	rw
PDO Mapping	No
Range	UNSIGNED16
Default value	0

Index	100Dh
Name	Life Time Factor
Object	VAR
Type	UNSIGNED8

Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	0

The 100Ch object allows programming the time necessary (in milliseconds) for a fault occurrence being detected, in case the SCA06 does not receive any telegram from the master. The 100Dh object indicates how many faults in sequence are necessary until it be considered that there was really a communication error. Therefore, the multiplication of these two values will result in the total necessary time for the communication error detection using this object. The value 0 (zero) disables this function.

Once configured, the SCA06 starts counting these times starting from the first *Node Guarding* telegram received from the network master. The master telegram is of the remote type, not having data bytes. The identifier is equal to 700h + Node-ID of the destination slave. However the slave response telegram has 1 data byte with the following structure:

Identifier	byte 1	
	bit 7	bit 6 ... bit 0
700h + Node-ID	Toggle	Slave state

This telegram has one single data byte. This byte contains, in the seven least significant bits, a value to indicate the slave state (4 = stopped, 5 = operational and 127 = preoperational), and in the eighth bit, a value that must be changed at every telegram sent by the slave (*toggle bit*).

If the servo drive SCA06 detects an error using this mechanism, it will turn automatically to the preoperational state and indicate alarm A135 on its HMI.

NOTE!
- This object is active even in the stopped state (see table 6.8).
- The value 0 (zero) in any of these two objects will disable this function.
- If after the error detection the service is enabled again, then the error indication will be removed from the HMI.
- The minimum value accepted by the SCA06 is 1ms., but considering the transmission rate and the number of nodes in the network, the times programmed for this function must be consistent, so that there is enough time for the transmission of the telegrams and also that the rest of the communication be able to be processed.
- For any every slave only one of the two services - Heartbeat or Node Guarding – can be enabled.

6.5.3 Error Control – Heartbeat

The error detection through the *Heartbeat* mechanism is done using two types of objects: the *Heartbeat* producer and the *Heartbeat* consumer. The producer is responsible for sending periodic telegrams to the network, simulating a heartbeat, indicating that the communication is active and without errors. One or more consumers can monitor these periodic telegrams, and if they cease occurring, it means that any communication problem occurred.

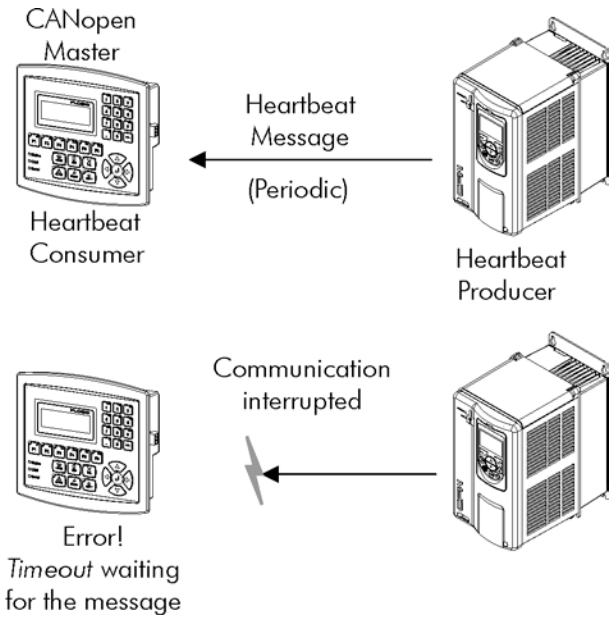


Figure 6.6: *Error control service – Heartbeat*

One device of the network can be both producer and consumer of *heartbeat* messages. For example, the network master can consume messages sent by a slave, making it possible to detect communication problems with the master, and simultaneously the slave can consume *heartbeat* messages sent by the master, also making it possible to the slave detect communication fault with the master.

The SCA06 has the producer and consumer of *heartbeat* services. As a consumer, it is possible to program up to 4 different producers to be monitored by the inverter.

Index	1016h
Name	Consumer Heartbeat Time
Object	ARRAY
Type	UNSIGNED32

Sub index	0
Description	Number of the last sub-index
Access	ro
PDO Mapping	No
Range	-
Default value	8

Sub index	1 – 8
Description	Consumer Heartbeat Time 1 – 8
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	0

At sub-indexes 1 to 8, it is possible to program the consumer by writing a value with the following format:

UNSIGNED32		
Reserved (8 bits)	Node-ID (8 bits)	Heartbeat time (16 bits)

- Node-ID: it allows programming the Node_ID for the *heartbeat* producer to be monitored.
- *Heartbeat time*: it allows programming the time, in 1 millisecond multiples, until the error detection if no message of the producer is received. The value 0 (zero) in this field disables the consumer.

Once configured, the *heartbeat* consumer initiates the monitoring after the reception of the first telegram sent by the producer. In case that an error is detected because the consumer stopped receiving messages from the *heartbeat* producer, the servo drive will turn automatically to the preoperational state and indicate alarm A135 in the HMI.

As a producer, the servo drive SCA06 has an object for the configuration of that service:

Index	1017h
Name	Producer Heartbeat Time
Object	VAR
Type	UNSIGNED16

Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	0

The 1017h object allows programming the time in milliseconds during which the producer has to send a *heartbeat* telegram to the network. Once programmed, the inverter initiates the transmission of messages with the following format:

Identifier	byte 1	
	bit 7	bit 6 ... bit 0
700h + Node-ID	Always 0	Slave state

NOTE!
- This object is active even in the stopped state (see table 6.8).
- The value 0 (zero) in the object will disable this function.
- If after the error detection the service is enabled again, then the error indication will be removed from the HMI.
- The time value programmed for the consumer must be higher than the programmed for the respective producer. Actually, it is recommended to program the consumer with a multiple of the value used for the producer.
- For any every slave only one of the two services - *Heartbeat* or *Node Guarding* – can be enabled.

6.6 INITIALIZATION PROCEDURE

Once the operation of the objects available for the servo drive SCA06 is known, then it becomes necessary to program the different objects to operate combined in the network. In a general manner, the procedure for the initialization of the objects in a CANopen network follows the description of the next flowchart:

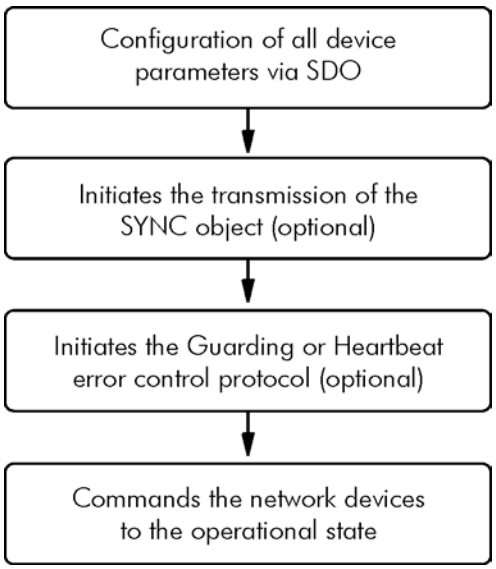


Figure 6.7: *Initialization process flowchart*

It is necessary to observe that the servo drive SCA06 communication objects (1000h to 1FFFh) are not stored in the nonvolatile memory. Therefore, every time the equipment is reset or switched off, it is necessary to redo the communication objects parameter setting. The manufacturer specific objects (starting from 2000h that represents the parameters), they are stored in the nonvolatile memory and, thus, could be set just once.

7 DESCRIPTION OF THE OBJECTS FOR DRIVES

The objects that are common for drives, defined by the CANopen specification in the CiA DSP 402 document, are described in this section. Regardless of the drive manufacturer, the objects mentioned here have a similar description and operation. This makes it easier the interaction and the interchangeability between different devices.

The figure 8 shows a diagram with the logic architecture and the operation of a drive through the CANopen network, with the different operation modes defined in this specification. Each operation mode has a set of objects that allows the configuration and operation of the drive in the network.

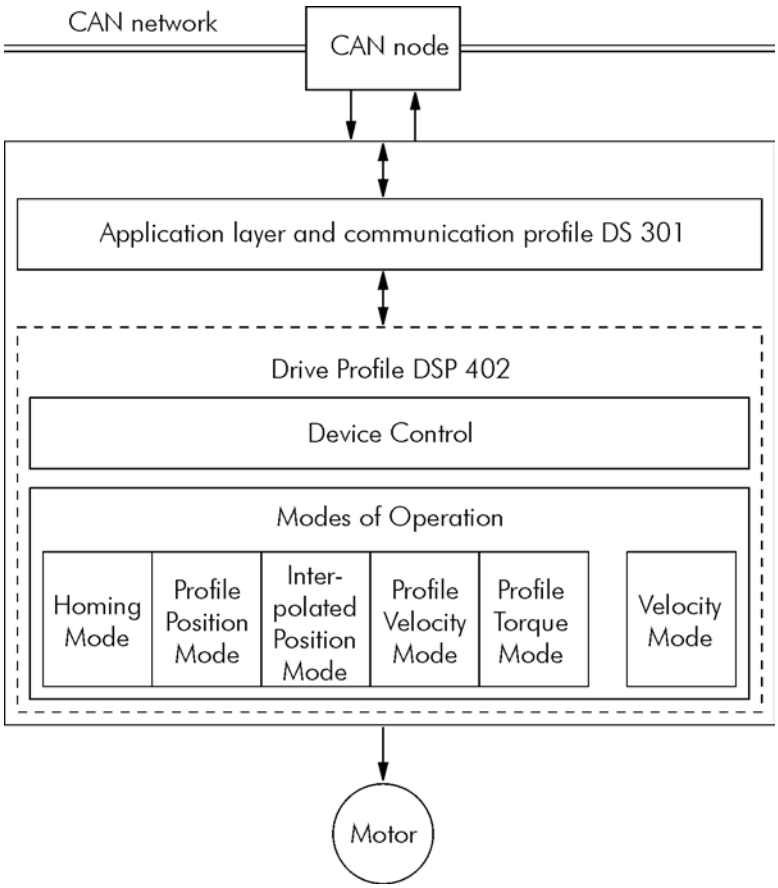


Figure 8: *Communication architecture for a drive in the CANopen network*

For the SCA06, only the *Velocity Mode* is supported. The following table presents the list of the available objects for the SCA06, divided according to the different operation modes of the inverter.

Table 10: *Object list – Drive Profile*

Index	Object	Name	Type	Access	PDO Mapping
Control Device					
6040h	VAR	ControlWord	Unsigned16	rw	Yes
6041h	VAR	StatusWord	Unsigned16	ro	Yes
6060h	VAR	Mode of operation	Integer8	rw	Yes
6061h	VAR	Modes of operation display	Integer8	ro	Yes
6502h	VAR	Supported drive modes	Unsigned32	ro	Yes
Factor Group					
608Fh	VAR	Position encoder resolution	Unsigned32	rw	No
6091h	VAR	Gear ration	Unsigned32	rw	No
6092h	VAR	Feed constant	Unsigned32	rw	No
Position Control Function					
6063h	VAR	Position actual value	Integer32	ro	Yes
6064h	VAR	Position actual value in user units	Integer32	ro	Yes
Profile Position Mode					
607Ah	VAR	Target position	Interger32	rw	Yes
6081h	VAR	Profile velocity	Unsigned32	rw	Yes
6083h	VAR	Profile acceleration	Unsigned32	rw	Yes
6084h	VAR	Profile deceleration	Unsigned32	rw	Yes
6086h	VAR	Motion profile type	Integer16	rw	Yes
Profile Velocity Profile					
6069h	VAR	Velocity sensor actual value	Integer32	ro	Yes
606Bh	VAR	Velocity demand value	Integer32	ro	Yes
606Ch	VAR	Velocity actual value	Integer32	ro	Yes
60FFh	VAR	Target velocity	Integer32	rw	Yes
Profile Torque Mode					
6071h	VAR	Target torque	Integer16	rw	Yes
6077h	VAR	Torque actual value	Integer16	ro	Yes
6087h	VAR	Torque slope	Unsigned32	rw	Yes
6088h	VAR	Torque profile type	Integer16	rw	Yes
Cyclic synchronous position mode/ Cyclic synchronous velocity mode					
60B1h	VAR	Velocity offset	Integer32	rw	Sim
60C2h	RECORD	Interpolation time period	Interpolation time period record	rw	Sim

Every time an object of that list is read or written, the SCA06 will map its functions in the inverter parameters. Thus, by operating the system through these objects, the value of the parameters can be changed according to the used function. In the next items a detailed description of each of these objects is presented, where the inverter parameters used to execute these object functions are indicated.

7.1 DEVICE CONTROL – OBJECTS FOR CONTROLLING THE DRIVE

Every drive operating in a CANopen network following the DSP 402 must be in accordance with the description of the following state machine:

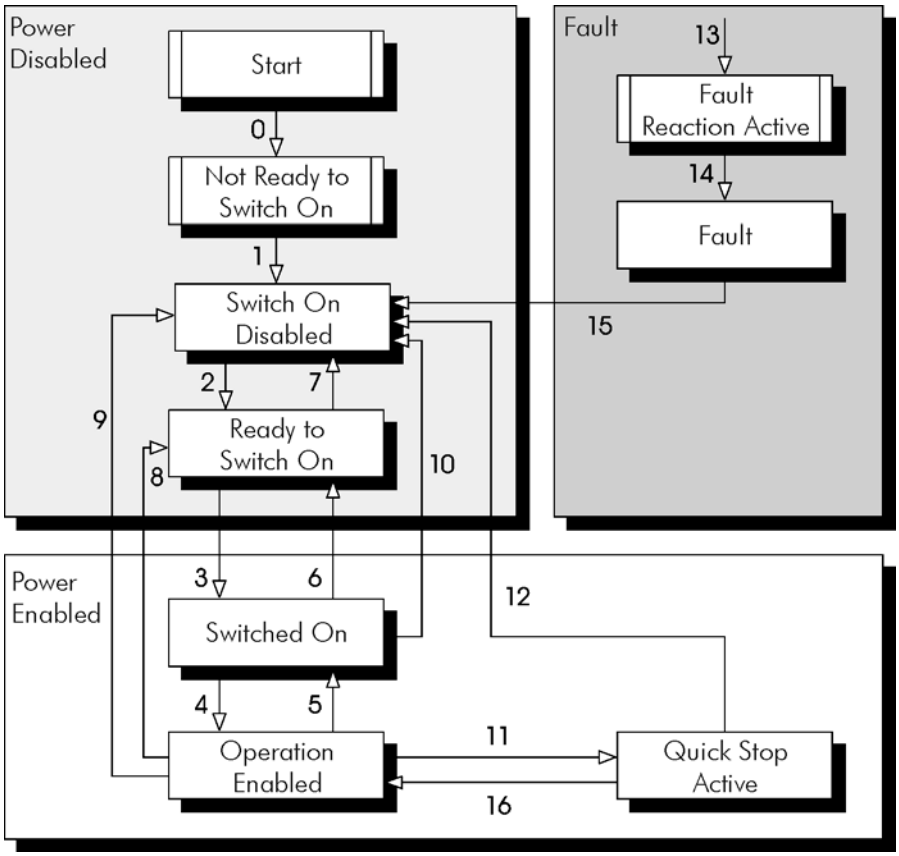


Figure 9: *State machine for drives*

Estate descriptions:

- **Not ready to switch on:** The inverter is initializing, it cannot be commanded.
- **Switch on disabled:** Initialization complete, the inverter is able to receive commands.
- **Ready to switch on:** Command to allow powering up the drive has been received.
- **Switched on:** Command for powering up the drive has been received.
- **Operation enabled:** The drive is enabled, being controlled according to the programmed operation mode. Power is being applied to the motor.
- **Quick stop active:** During the operation, the quick stop command was received. Power is being applied to the motor.
- **Fault reaction active:** A fault has occurred and the drive is performing the action related to the type of fault.
- **Fault:** Drive with fault. Disabled function, without power being applied to the motor.

NOTE!
The frequency inverter SCA06 does not have a switch for disabling / enabling the power section supply of the equipment. Therefore, the states described in the Power disabled group were implemented for a matter of compatibility with the described state machine; however the power section supply remains active even in these states.

Description of the transitions:

- **Transition 0:** The drive is switched on and the initialization procedure starts. The power section supply of the drive is active.
- **Transition 1:** Initialization completed (automatic).
- **Transition 2:** The *Shutdown* command has been received. The state transition is performed, but no action is taken by the SCA06.
- **Transition 3:** The *Switch on* command has been received. The state transition is performed, but no action is taken by the SCA06.
- **Transition 4:** The *Enable operation* command has been received. the servo drive is enabled.
- **Transition 5:** The *Disable operation* command has been received. the servo drive is disabled.

- **Transition 6:** The *Shutdown* command has been received. The state transition is performed, but no action is taken by the SCA06.
- **Transition 7:** The *Quick stop* and *Disable voltage* commands have been received. The state transition is performed, but no action is taken by the SCA06.
- **Transition 8:** The *Shutdown* command has been received. During the operation of the servo drive it is disabled, blocking the supply for the motor.
- **Transition 9:** The *Shutdown* command has been received. During the operation of the servo drive it is disabled, blocking the supply for the motor.
- **Transition 10:** The *Quick stop* or *Disable voltage* command has been received. The state transition is performed, but no action is taken by the SCA06.
- **Transition 11:** The *Quick stop* command has been received. the servo drive performs the stopping via ramp function.
- **Transition 12:** The *Disable voltage* command has been received. the servo drive is disabled.
- **Transition 13:** A fault is detected and the servo drive is disabled.
- **Transition 14:** After disabling the drive, it goes to the fault state (automatic).
- **Transition 15:** The *Fault reset* command has been received. the servo drive performs the fault reset and returns to the disabled and without fault state.
- **Transition 16:** The *Enable operation* command has been received. the servo drive performs the start via ramp function.

This state machine is controlled by the 6040h object, and the other states can be monitored by the 6041h object. Both objects are described next.

7.1.1 Object 6040h – Controlword

It controls the servo drive state

Index	6040h
Name	Controlword
Object	VAR
Type	UNSIGNED16
Used parameter	P0684

Access	rw
PDO Mapping	Yes
Range	UNSIGNED16
Default value	-

The bits of this word have the following functions:

15 – 9	8	7	6 – 4	3	2	1	0
Reserved	Halt	Fault reset	Operation mode specific	Enable operation	Quick stop	Enable voltage	Switch on

The bits 0, 1, 2, 3 and 7 allow controlling the drive state machine. The commands for state transitions are given by means of the bit combinations indicated in the table 11. The bits marked with “x” are irrelevant for the command execution.

Table 11: *Control word commands*

Command	Control word bits					Transitios
	Fault reset	Enable operation	Quick stop	Enable voltage	Switch on	
Shutdown	0	×	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Disable voltage	0	×	×	0	×	7, 9, 10, 12
Quick stop	0	×	0	1	×	7, 10, 11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4, 16
Fault reset	0 → 1	×	×	×	×	15

The bits 4, 5, 6 and 8 have different functions according to the used operation mode.

NOTE!
For the commands sent by the control word to be executed by the servo drive SCA06, it is necessary that the drive be programmed for the "CANopen" mode of operation. This programming is done on parameter P0202.

7.1.2 Object 6041h – Statusword

It indicates the SCA06 present state.

Index	6041h
Name	Statusword
Object	VAR
Type	UNSIGNED16
Used parameter	P0680

Access	ro
PDO Mapping	Yes
Range	UNSIGNED16
Default value	-

Table 12: *Statusword bit function*

Bit	Description
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
8	Reserved
9	Remote
10	Target reached
11	Internal limit active
12 – 13	Operation mode specific
14 – 15	Reserved

In this word the bits 0, 1, 2, 3, 5 and 6 indicate the state of the device according to the state machine described in the figure 9. The table 13 describes the combinations of these bits for the state indications. The bits marked with "x" are irrelevant for the state indication.

Table 13: *Drive states indicated through the Statusword*

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

The other bits indicate a specific condition for the drive.

- Bit 4 – Voltage enabled: indicates that the drive power section is being fed.
- Bit 7 – Warning: It is not used for the SCA06.
- Bit 9 – Remote: indicates when the drive is in the remote mode and accepts commands via the CANopen[7] network.
- Bit 10 – Target reached: indicates when the drive is operating at the reference value, which depends on the used operation mode. It is also set to 1 when the functions Quick stop or Halt are activated.

[7] It depends on the inverter programming.

- Bit 11 – Internal limit active: not is used for the servo drive SCA06.
- Bits 12 and 13 – Operation mode specific: they depend on the drive operation mode.

7.1.3 Object 6060h – Modes of Operation

It allows programming the SCA06 operation mode.

Index	6060h
Name	Modes of operation
Object	VAR
Type	INTEGER8
Used parameter	-

Access	rw
PDO Mapping	Yes
Range	INTEGER8
Default value	-

Acceptable values for this object are described in table 14. Other values are reserved.

Table 14: *Modes of operation for servo drive SCA06*

Value	Modes of Operation
1	Profile Position Mode
3	Profile Velocity Mode
4	Profile Torque Mode
8	Cyclic sync position mode
9	Cyclic sync velocity mode

7.1.4 Object 6061h – Modes of Operation Display

It indicates the SCA06 operation mode.

Index	6061h
Name	Modes of operation display
Object	VAR
Type	INTEGER8
Used parameter	-

Access	rw
PDO Mapping	Yes
Range	INTEGER8
Default value	-

The value presented at this object follows the same options for object 6060h.

7.1.5 Objeto 6502h – Supported Drive Modes

It indicates the modes of operation supported by the drive. Each bit represents a mode of operation, and the value 1 in bit indicates that the mode of operation is supported.

31 – 15	15 – 7	9	8	7	6	5	4	3	2	1	0
Manufecturer specific	reserved	cst	csv	csp	ip	hm	reserved	tq	pv	vl	PP

The SCA06 servo drive features three modes of operation:

- pp: Profile Position mode.
- pv: Profile Velocity mode.
- tq: Torque mode.
- csv: Cyclic sync velocity mode
- csp: Cyclic sync position mode

Knowing the modes supported on the SCA06, the value of 0Dh for this object is defined.

Index	6502h
Name	Supported drive modes
Object	VAR
Type	UNSIGNED32

Sub-index	0
Access	ro
Mappable	Yes
Range	UNSIGNED8
Default Value	0Dh

7.2 FACTOR GROUP – OBJECTS FOR UNIT CONVERSION

This object group allows converting units for objects that represent position values. These values will have their scale and dimension defined according to the programmed notation and dimension values, as described below:

7.2.1 Object 608Fh – Position Encoder Resolution

This object defines the increment of the encoder according to the motor speed:

Position encoder resolution = encoder increments / motor revolutions

Index	608Fh
Name	Position encoder resolution
Object	ARRAY
Type	UNSIGNED32

Sub-index	0
Description	Number of the last sub-index
Access	ro
Mappable	No
Range	UNSIGNED8
Default Value	02h

Sub-index	1
Description	Encoder increments
Access	rw
Mappable	No
Range	UNSIGNED32
Default Value	FFh

Sub-index	2
Description	Motor revolutions
Access	rw
Mappable	No
Range	UNSIGNED32
Default Value	01h

Possible values for the sub-index 1 (Encoder increments):

Table 15: *Values for the Encoder Increments Sub-index*

Value	Mode of operation
41	Degrees
42	Minutes
43	Seconds
FF	Internal unit - 65536 increments by revolution

The sub-index 2 (Motor revolutions) only accepts value equal to 1.

7.2.2 Object 6091h – Gear Ratio

This object indicates the configuration and number of motor shaft revolutions and number of driving shaft revolutions, that is, it defines the gear ratio. The gear ration is defined by the following formula:

Gear ratio = motor shaft revolutions / driving shaft revolutions

Index	6091h
Name	Gear ratio
Object	ARRAY
Type	UNSIGNED32

Sub-index	0
Description	Number of the last sub-index
Access	ro
Mappable	No
Range	UNSIGNED8
Default Value	02h

Sub-index	1
Description	Motor revolutions
Access	rw
Mappable	No
Range	UNSIGNED32
Default Value	01h

Sub-index	2
Description	Shaft revolutions
Access	rw
Mappable	No
Range	UNSIGNED32
Default Value	01h

The only possible value for the sub-index 1 and sub-index 2 is 1.

7.2.3 Object 6092h – Feed Constant

This object indicates the distance per one revolution of the motor shaft.

Index	6092h
Name	Feed constant
Object	ARRAY
Type	UNSIGNED32

Sub-index	0
Description	Number of the last sub-index
Access	ro
Mappable	No
Range	UNSIGNED8
Default Value	02h

Sub-index	1
Description	Feed
Access	rw
Mappable	No
Range	UNSIGNED32
Default Value	FFh

Sub-index	2
Description	Shaft revolutions
Access	rw
Mappable	No
Range	UNSIGNED32
Default Value	01h

Possible values for the sub-index 1 (Feed):

Table 16: Values for the Feed Sub-index

Value	Mode of operation
41	Degrees
42	Minutes
43	Seconds
FF	Internal unit - 65536 increments by revolution

The sub-index 2 (Shaft revolutions) only accepts value equal to 1.

7.3 POSITION CONTROL FUNCTION – POSITION CONTROLLER

This object group is used to describe the operation of the position controller in closed loop.

7.3.1 Object 6063h – Position Actual Value

It represents the actual position of the motor shaft in increments. A complete revolution represents 65536 increments.

Index	6063h
Name	Position actual value
Object	VAR
Type	INTEGER32

Sub-index	0
Access	ro
Mappable	Yes
Range	INTEGER32
Default Value	-

The value of this object always represents the shaft position in a single revolution only. The number of revolutions is not controlled by this object.

7.3.2 Object 6064h – Position Actual Value in User Units

It represents the actual position of the motor shaft. The value of this object can be transformed from internal units to user-defined values, according to the settings of the objects 608Fh, 6091h and 6092h, as per Table 17.

Index	6064h
Name	Position actual value
Object	VAR
Type	INTEGER32

Sub-index	0
Access	ro
Mappable	Yes
Range	INTEGER32
Default Value	-

The value of this object always represents the shaft position in a single revolution only. The number of revolutions is not controlled by this object.

Table 17: Programming of the Factor Group objects

Object / Unit	608Fh sub-index 1	608Fh sub-index 2	6091h sub-index 1	6091h sub-index 2	6092h sub-index 1	6092h sub-index 2
Degrees	41h	1	1	1	41h	1
Minutes	42h	1	1	1	42h	1
Seconds	43h	1	1	1	43h	1
Internal unit	FFh	1	1	1	FFh	1

7.4 PROFILE POSITION MODE – OBJECTS FOR DRIVE CONTROL

This mode of operation allows the control of the SCA06 servo drive by adjusting of the position set-point, which can be executed following two methods:

- single set-point.
- set of set-points.

Regardless of the used method, the following objects must be configured:

- 0x6081 – Profile Velocity;
- 0x6083 – Profile Acceleration;
- 0x6084 – Profile Deceleration;
- 0x6086 – Motion Profile Type;
- 0x607A – Target Position;

After adjusting the speed, acceleration and set-point, the following procedure must be executed:

Enable the drive by writing 15 on the object 0x6041 - ControlWord;
Write on the bits 9, 8, 6, 5 and 4 of the object 0x6041 – ControlWord, according to Table 18and Table 19;

The execution status of the positioning can be checked in the object 0x6040 – StatusWord according to Table 20. The SET-POINT ACKNOWLEDGE bit on the status object (StatusWord – 6040h) will be set, indicating that a new set-point was received. If the set-point is accepted, the bit is reset. When the set-point is reached, the TRAGET REACHED bit on the status object will be set. Figure 10 shows an example of set-point writing.

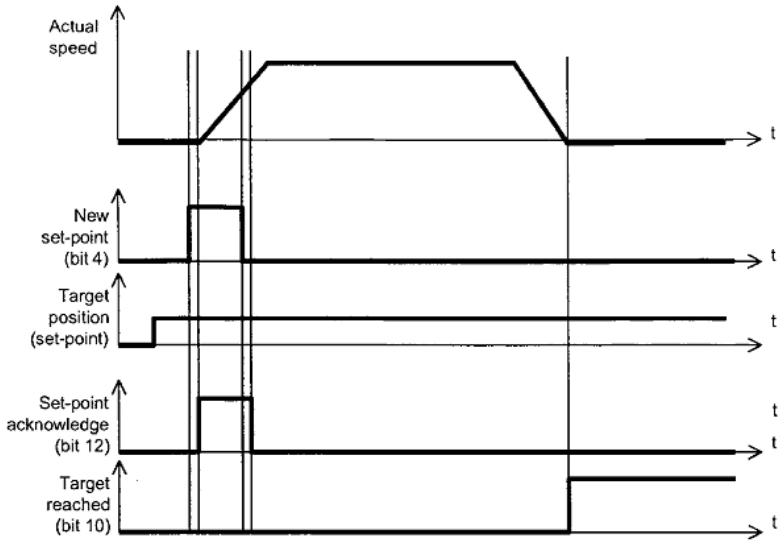


Figure 10: *Adjustment of the position set-point (Source: IEC 61800-7-201)*

Single set-point

The single set-point method is used when you want to execute a new set-point immediately. Figure 11 illustrates the method.

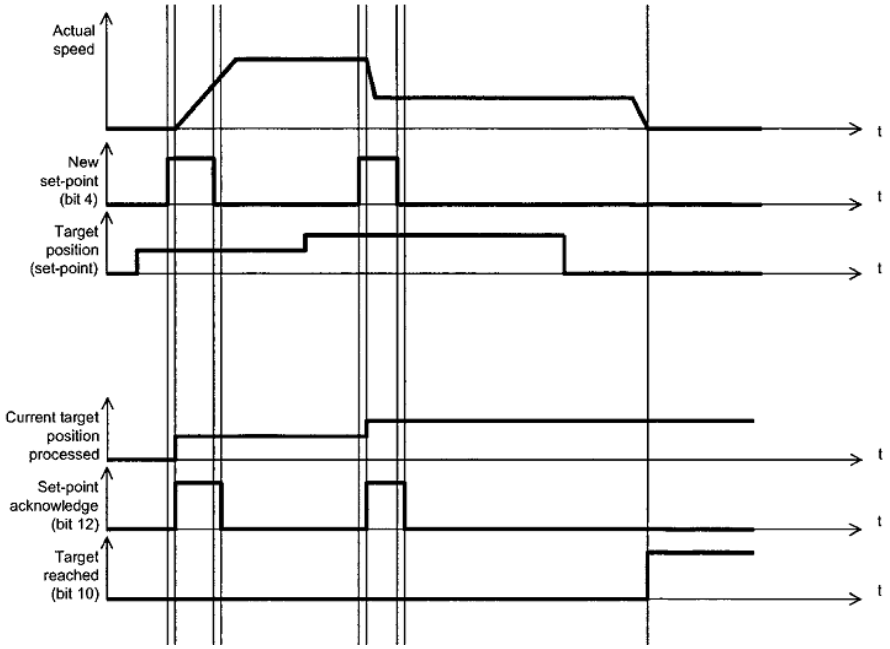


Figure 11: *Single set-point method (Source: IEC 61800-7-201)*

Set of set-points.

The set of set-point method is used when you want to execute a new set-point only after the completion of the previous one. Figure 12 illustrates the method.

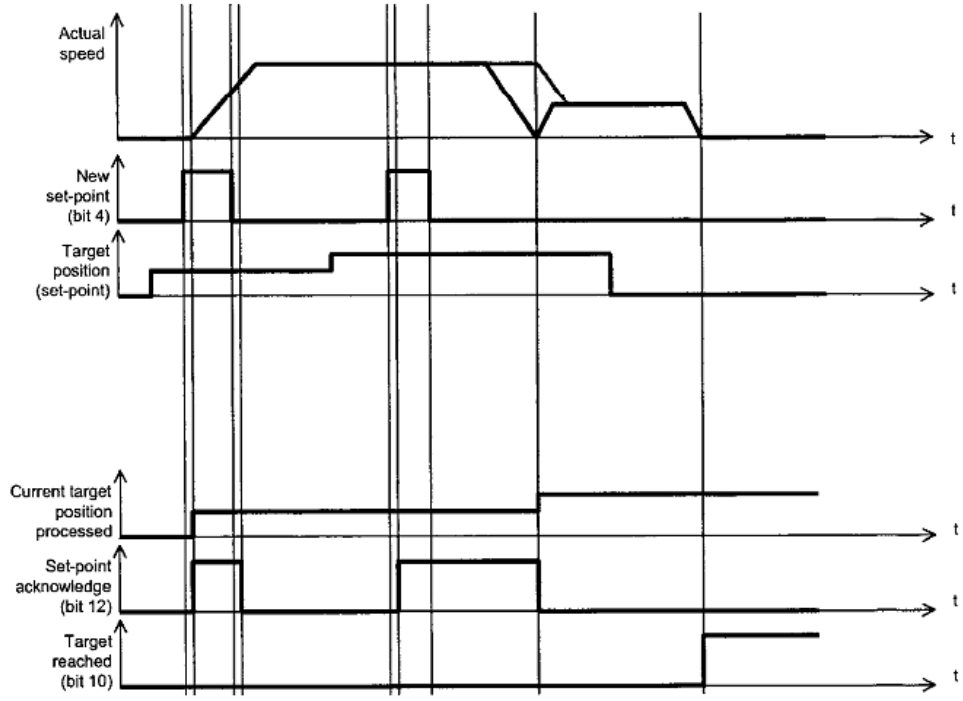


Figure 12: *Set of set-point method (Source: IEC 61800-7-201)*

The servo drive SCA06 can store two set-points, the one which is in execution and the one that will be executed, as illustrated in figure 13.

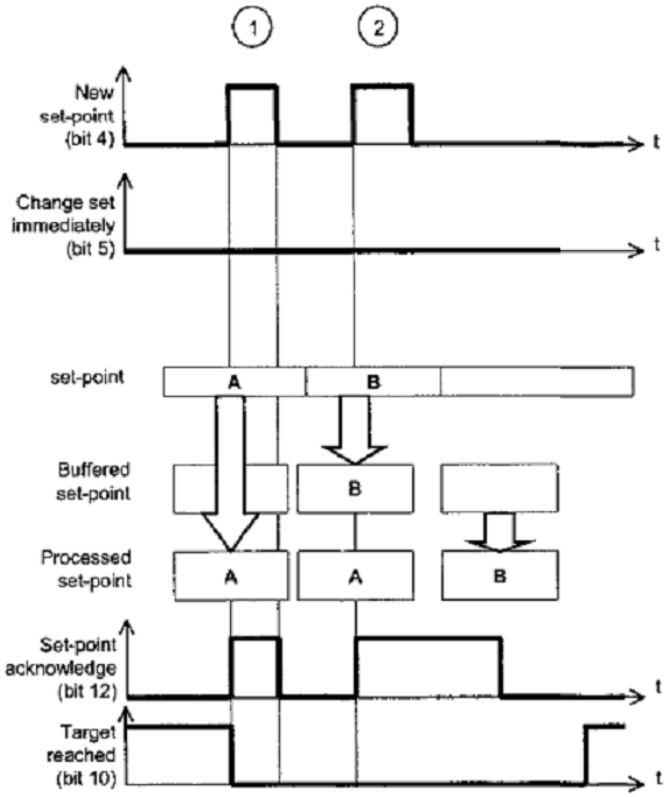


Figure 13: *Storage of set-point (Source: IEC 61800-7-201)*

7.4.1 Control and Status Bits

The profile mode position uses some bits of the ControlWord and StatusWord objects to control and monitor its operation. For the ControlWorld object the following bits are used:

Bit 4 – New set-point;
Bit 5 – Change set immediately;
Bit 6 – absolute (0) /relative (1);
Bit 8 – Halt (not implemented in the SCA06);
Bit 9 – Change on set-point.

Table 18 and table 19 contain the definition of the control bits.

Table 18: *Positioning Mode – definition of the bits 4, 5 and 9*

Bit 9	Bit 5	Bit 4	Definition
0	0	0 → 1	Position must be completed before the next one starts.
X	1	0 → 1	Next position must start immediately.
1	0	0 → 1	Option not implemented on the SCA06

Table 19: *Positioning Mode – definition of bits 6 and 8*

Bit	Value	Definition
6	0	Position reference must be an absolute value.
	1	Position reference must be a relative value.
8	0	Positioning must be executed or continued.
	1	Shaft must be stopped according to object 605Dh.

For the StatusWord object, the following bits are used:

Bit 10 – Target reached;
Bit 12 – Set-point acknowledge;
Bit 13 – Following error.

Table 20 contains the definition of the status bits.

Table 20: *Positioning Mode – definition of bits 10, 12 and 13*

Bit	Value	Definition
10	0	Position reference not reached.
	1	Position reference reached.
12	0	Previous position reference already processed, waiting for new position reference.
	1	Previous position reference in process, replacement of position reference will be accepted.
13	0	No Following error
	1	Following error

7.4.2 Object 607Ah – Target Position

It allows programming the position reference for the servo drive SCA06 in positioning mode. The 16 most relevant bits inform the number of revolutions and the 16 bits least relevant ones inform the fraction of revolution. The scale used in this object is 65536 for number of revolutions and 65536 increments for one revolution of the shaft. The value of this object must be interpreted as absolute or relative, according to the status of Bit 6 of the ControlWord object.

Number of revolution 16MSB	Fraction of revolution 16LSB

Index	607Ah
Name	Target position
Object	VAR
Type	INTEGER32

Sub-index	0
Access	rw
Mappable	Yes
Range	INTEGER32
Default Value	0000 0000h

7.4.3 Object 6081h – Profile Velocity

It allows programming the speed normally reached at the end of the acceleration ramp during a movement profile. The value set in this object must be between 0 and 9999 rpm.

Index	6081h
Name	Profile Velocity
Object	VAR
Type	UNSIGNED32

Sub-index	0
Access	rw
Mappable	Yes
Range	UNSIGNED32
Default Value	0000 0000h

7.4.4 Object 6083h – Profile Acceleration

It allows programming the acceleration ramp until the motor shaft reaches the programmed speed. The scale used is the ms/krpm scale and the values must be between 1 and 32767.

Index	6083h
Name	Profile Acceleration
Object	VAR
Type	UNSIGNED32

Sub-index	0
Access	rw
Mappable	Yes
Range	UNSIGNED32
Default Value	0000 0001h

7.4.5 Object 6084h – Profile Deceleration

It allows programming the acceleration ramp until the motor shaft reaches the zero speed. The scale used in this object is the same as that of the object 6083h.

Index	6084h
Name	Profile deceleration
Object	VAR
Type	UNSIGNED32

Sub-index	0
Access	rw
Mappable	Yes
Range	UNSIGNED32
Default Value	0000 0001h

7.4.6 Object 6086h – Motion Profile Type

It allows programming the profile of the acceleration and deceleration ramp for the drive.

Index	6086h
Name	Motion profile type
Object	VAR
Type	INTERGER32

Sub-index	0
Access	rw
Mappable	Yes
Range	INTERGER16
Default Value	FFFFh

Possible values for this object:

Table 21: *Values for the Motion Profile Type Sub-index*

Value	Definition
0000h	Linear ramp
FFFFh	No ramp

7.5 PROFILE VELOCITY MODE – OBJECTS FOR DRIVE CONTROL

This mode of operation allows controlling the drive in a simple way, providing functions, such as:

- Calculation of the reference value.
- Speed capture and monitoring.
- Speed limitation.
- Speed ramps, among other functions.

Those functions are executed based on a set of objects for the configuration of this mode of operation.

7.5.1 Control and Status Bits

Bits 4, 5, 6 and 8 of the control word (object 6040h – Controlword) have the following functions in the speed mode:

Table 22: *Speed Mode – definition of bits 4, 5, 6 and 8*

Bit	Name	Value	Description
4	Reserved		
5	Reserved		
6	Reserved		
8	Halt	0	Executes movement
		1	Stops shaft

For the StatusWord object, the following bits are used:

Bit 10 – Target reached;
Bit 12 – Speed;
Bit 13 – Max slippage error (not implemented).

Table 23: *Speed Mode – definition of bits 10, 12 and 13*

Bit	Value	Definition
10	0	Halt = 0 - speed reference not reached. Halt = 1 - speed different from 0 (zero)
	1	Halt = 0 - speed reference reached. Halt = 1 - speed equal to 0 (zero)
12	0	Speed different from 0 (zero)
	1	Speed equal to 0 (zero)
13	0	Not implemented
	1	

7.5.2 Object 6069h – Velocity Sensor Actual Value

It allows the reading of the sensor used to measure the motor speed. The servo drive SCA06 uses a solve as position (the angular speed is obtained by deriving this value in time), so the sensor provides a value proportional to the angular position. The sensor has resolution of 14 bits, and one complete revolution provides 16384 different position values.

Index	6069h
Name	Velocity sensor actual value
Object	VAR
Type	INTEGER32

Sub-index	0
Access	ro
Mappable	Yes
Range	INTERGER32
Default Value	-

7.5.3 Object 606Bh – Velocity Demand Value

It indicates the speed provided by the trajectory generator of the servo drive, used by the speed controller to control the motor. The value provided by this object is given in the internal scale of the SCA06, where 0x7FFF FFFF → 10.000 rpm.

Index	606Bh
Name	Velocity demand value
Object	VAR
Type	INTEGER32

Sub-index	0
Access	ro
Mappable	Yes
Range	INTERGER32
Default Value	-
Minimum value	0x8000 0001
Maximum value	0x7FFF FFFF

7.5.4 Object 606Ch – Velocity Actual Value

It indicates the motor speed. The value provided by this object is given in the internal scale of the SCA06, where 0x7FFF FFFF → 10.000 rpm.

Index	606Ch
Name	Velocity actual value
Object	VAR
Type	INTEGER32

Sub-index	0
Access	ro
Mappable	Yes
Range	INTERGER32
Default Value	-

7.5.5 Object 60FFh – Target Velocity

Allows programming the speed reference for the servo drive SCA06 in speed mode. The value set in this object must observe the internal scale of the SCA06, where 0x7FFF FFFF → 10.000 rpm and 0x8000 0000 → -10.000 rpm

Index	60FFh
Name	Target velocity
Object	VAR
Type	INTERGER32

Sub-index	0
Access	rw
Mappable	Yes
Range	INTERGER32
Default Value	0000 0000h

7.6 PROFILE TORQUE MODE – OBJECTS FOR DRIVE CONTROL

This mode allows controlling the by means of a torque reference received by the CANopen network.

Those functions are executed based on a set of objects for the configuration of this mode of operation.

7.6.1 Control and Status Bits

Bits 4, 5, 6 and 8 of the control word (object 6040h – Controlword) have the following functions in the speed mode:

Table 24: *Torque Mode – definition of the bits 4, 5, 6 and 8*

Bit	Name	Value	Description
4	Reserved		
5	Reserved		
6	Reserved		
8	Halt	0	Executes movement
		1	Stops shaft

For the StatusWord object, the following bits are used:

Bit 10 – Target reached;
Bit 12 – Reserved;
Bit 13 – Reserved.

Table 25: *Torque Mode – definition of bits 10,12 and13*

Bit	Value	Definition
10	0	Torque reference not reached.
	1	Torque reference reached.
12	0	Reserved
	1	
13	0	Reserved
	1	

7.6.2 Object 6071h – Target Torque

It allows programming the torque reference for the servo drive SCA06 in the torque mode. The scale used to write on this object is provided in parts per thousand of the motor rated torque.

Index	6071h
Name	Target Torque
Object	VAR
Type	INTERGER16

Sub-index	0
Access	rw
Mappable	Yes
Range	INTERGER16
Default Value	0000h

7.6.3 Object 6077h – Torque Actual Value

It indicates the actual motor torque. The value is provided in part per thousand of the rated motor torque.

Index	6077h
Name	Torque actual value
Object	VAR
Type	INTERGER16

Sub-index	0
Access	rw
Mappable	Yes
Range	INTERGER16
Default Value	0

7.6.4 Object 6087h – Torque Slope

It allows programming the rate of torque variation in time (torque ramp) for the servo drive SCA06. The scale used is of parts per thousand of the rated motor torque per second.

Index	6087h
Name	Torque slope
Object	VAR
Type	UNSIGNED32

Sub-index	0
Access	rw
Mappable	Yes
Range	UNSIGNED32
Default Value	0

7.6.5 Object 6088h – Torque Profile Type

Is used to select the type of torque profile.

Index	6088h
Name	Torque Profile type
Object	VAR
Type	INTERGER16

Sub-index	0
Access	rw
Mappable	Yes
Range	INTERGER16
Default Value	FFFFh

Possible values for this object:

Table 26: *Values for the Torque Profile Type Sub-index*

Value	Definition
0000h	Linear ramp of the torque
FFFFh	No ramp

7.7 CYCLIC SYNCHRONOUS POSITION MODE

In cyclic synchronous manner, it provides a target position to the drive device, wich performs position control, velocity control and torque control. The overall structure for this mode is shown in Figure 14.

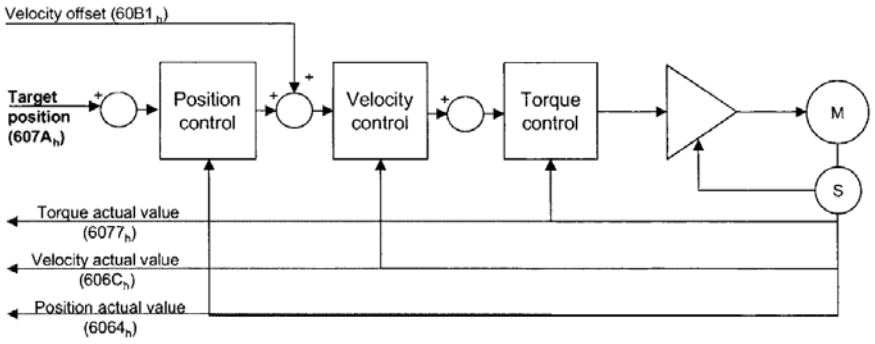


Figure 14: *Cyclic synchronous position mode overview.*

7.7.1 Control and status Bits

This mode uses no mode specific bits of the contolword and three bits of the statusword for mode-specific purposes. Table 27 defines the values for bit 10, 12 and 13 of the statusword.

Table 27: *definition of bit 10, bit 12 and bit 13*

Bit	Value	Definition
10	0	Reserved.
	1	Reserved
12	0	Target position is ignored
	1	Target position shall be used as input to position control loop
13	0	No following error
	1	Following error

7.7.2 Object 60B1h – Velocity Offset

This object provide the offset for the velocity value. The offset shall be given in user defined velocity units. This object contains the input value of velocity feed forward.

Index	60B1h
Name	Torque Profile type
Object	VAR
Type	Integer32

Sub-index	0
Access	rw
Mappable	Yes
Range	Integer32
Default Value	0h

7.7.3 Objeto 60C2h – Interpolation time period

This object shall indicate the configured interpolation cycle time.

Index	60C2h
Name	Interpolation time period
Object	Record
Type	Interpolation time period record

Sub-index	0
Description	Highest sub-index supported
Access	ro
Mappable	Não
Range	02h
Default Value	02h

Sub-index	1
Description	Interpolation time period value
Access	rw
Mappable	Não
Range	UNSIGNED8
Default Value	01h

Sub-index	2
Description	Interpolation time index
Access	Rw
Mappable	Não
Range	-128 a 63
Default Value	-3

7.7.4 Mode configuration

The objects below will be configured for the drive works in Cyclic Synchronization Position Mode:

- 0x6040 - Controlword
- 0x6060 – Modes of Operation
- 0x60C2 – Interpolation Time Type
- 0x60B1 – Velocity Offset
- 0x6086 – Motion Profile Type
- 0x607A – Target Position;

7.8 CYCLIC SYNCHRONOUS VELOCITY MODE

In cyclic synchronous manner, it provides a target velocity to the drive device, wich performs velocity control and torque control. The overall structure for this mode is shown in Figure 15.

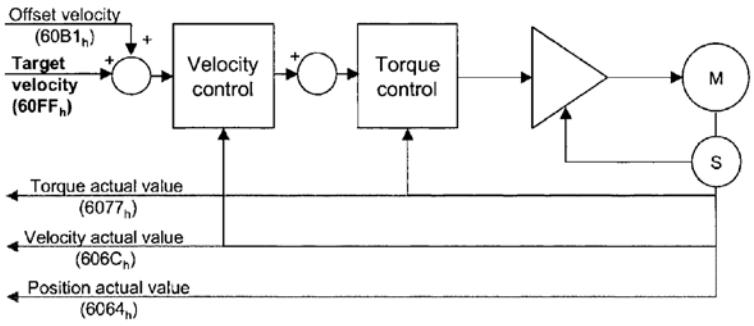


Figure 15: *Cyclic synchronous velocity mode overview.*

7.8.1 Control and Status Bits

This mode uses no mode specific bits of the contolword and three bits of the statusword for mode-specific purposes. Table 28 defines the values for bit 10, 12 and 13 of the statusword.

Table 28: definition of bit 10, bit 12 and bit 13

Bit	Value	Definition
10	0	Reserved.
	1	Reserved
12	0	Target velocity ignored
	1	Target velocity shall be used as input to velocity control loop
13	0	Reserved
	1	Reserved

7.8.2 Object 60B1h – Velocity Offset

In cyclic synchronous velocity mode, it contains the command offset of the drive device.

Index	60B1h
Name	Torque Profile type
Object	VAR
Type	Integer32

Sub-index	0
Access	rw
Mappable	Yes
Range	Integer32
Default Value	0h

7.8.3 Object 60C2h – Interpolation time period

According of item 7.7.3.

7.8.4 Mode configuration

The objects below will be configured for the drive works in Cyclic Synchronization Velocity Mode:

- 0x6040 - Controlword
- 0x6060 – Modes of Operation
- 0x60C2 – Interpolation Time Type
- 0x60B1 – Velocity Offset
- 0x6086 – Motion Profile Type
- 0x60FF – Target Velocity;

8 OPERATION IN CANOPEN NETWORK – MASTER MODE

In addition to operating as a salve, the servo drive SCA06 can also operate as master of the CANopen network. Below are described the characteristics and functions of the SCA06 as master of the CANopen network.

8.1 ENABLING OF THE MASTER CANOPEN FUNCTION

As default, the servo drive SCA06 is programmed to operate as slave of the CANopen network. The programming of the equipment as network master must be done by using the WSCAN software, which also allows the configuration of the entire CANopen network. The detailed description of the windows and functions of the WSCAN software is obtained in the "Help" menu of the software itself.

After the configuration of the master is ready, it is necessary to download[8] the configurations via one of the programming interfaces of the product – refer to the user's manual for further information. Once set as network master, if necessary to erase those configurations, the function to erase the user's program – trough P00204 – also erases the configurations of the CANopen master.

NOTE!
The CANopen network is a flexible network that allows several forms of configurations and operation. However, in order to use this flexibility, it is necessary that the user know well both the communication functions and objects used to configure the network, and the WSCAN programming software.

8.2 CHARACTERISTICS OF THE CANOPEN MASTER

The servo drive SCA06 allows controlling a group of up to 8 slaves, using the following communication services and resources:

- Network manager task (NMT)
- 8 transmission PDOs
- 8 reception PDOs
- 8 Heartbeat Consumers
- Heartbeat Producer
- SDO Client
- SYNC producer/consumer
- Mapping in the PDOs made by using user's parameters

The physical characteristics – installation, connector, cable, etc. – are the same for the SCA06 operating as both master and slave. The configurations of address and baud rate are also necessary to operate as master, but these configurations are programmed by the WSCAN software according to the properties defined for the master in the software itself.

8.3 OPERATION OF THE MASTER

Once programmed to operate as master, the servo drive SCA06 will execute the following steps to initialize, in a sequence, each slave:

- 1st: send the communication reset command to the entire network, so that the slaves initialize with known values for the communication objects.
- 2nd: Identification of the equipment in network, trough the reading via SDO of the object 1000h/00h – Object Identification.
- 3rd: Writing via SDO of all the objects programmed for the slave, which usually includes the configuration and mapping of the TPDOs and RPDOs, node guarding, heartbeat, besides the specific objects of the manufacturer, in case they are programmed.
- 4th: Start the error control task – node guarding or heartbeat – if they are programmed.
- 5th: send the slave to mode of operation.

8 During the download of the configurations, the CANopen communication will be disabled, and it will be enabled again at the end of the operation.

If one of these steps fails, the error of communication with slave will occur. Depending of the configurations, the slave initialization will be aborted, and the master will initialize the next slave, returning to the slave with error after trying to initialize all the other network slaves.

Similarly, if, during the operation of a slave, an error is identified in the error control task, depending on the configurations of the master, the slave will be automatically reset and the initialization procedure will be run again.

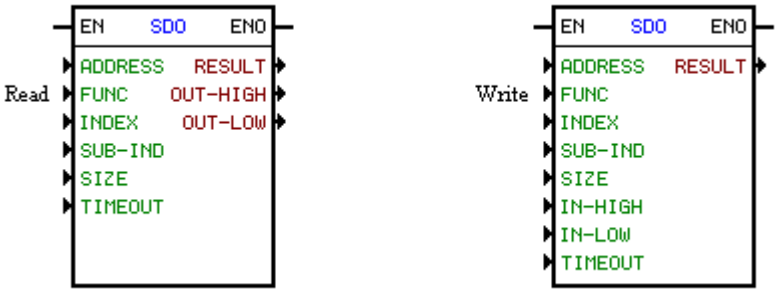
NOTE!
The communication status and the status of each slave can be observed in system markers.

8.4 BLOCKS FOR THE CANOPEN MASTER

In addition to the communication objects and the configurations made on the WSCAN software, blocks for monitoring and sending commands are also available. They can be used during the preparation of the ladder application for the servo drive SCA06. It is not necessary to use these blocks during the equipment operation, but they provides more flexibility and simplify the communication troubleshooting during the operation of the servo drive SCA06.

8.4.1 CANopen SDO – Data Reading/Writing via SDO

Block for data reading or writing via SDO of a remote slave. It allows reading or writing objects in network with size up to 4 bytes.



DESCRIPTION:

It consists of one EN input, one ENO output and 9 arguments, which are:

- ADDRESS : Address of the CANopen network node
- FUNC : Function (reading or writing)
- INDEX : Object index which you wish to read or write (decimal)
- SUB-IND : Object sub-index which you wish to read or write (decimal)
- SIZE : Object size which you wish to read or write (bytes)
- TIMEOUT : Waiting time in ms to read or write the value
- RESULT : Result of the execution of the block
  - 0 = successfully executed
  - 1 = card cannot execute the function (example: master not enabled)
  - 2 = timeout in the waiting for the response of the master
  - 3 = slave returned error
- OUT-HIGH : Most significant value of the object read (word)
- OUT-LOW : Least significant value of the object read (word)
- IN-HIGH : Most significant value to be written on the object (word)
- OUT-HIGH : Least significant value to be written on the object (word)

The EN input is responsible for enabling the block.
The ENO output goes to 1 after executing the block

OPERATION:

If the EN input is zero, the block is not executed.
If the EN input undergoes a transition from 0 to 1, the card sends a message via CANopen network to a network slave, according to the programmed arguments. If the block is programmed for reading, the card will make a request to the slave, and the value reported by the slave will be saved on the output arguments. If the block is programmed for writing, the input arguments are written to the corresponding object of the slave. After the execution of the block, the ENO output goes to 1 and will only return to zero after the EN input goes to zero.

9 SYSTEM MARKERS FOR CAN/CANOPEN

For CAN interface and CANopen communication, the following reading markers (%RS) and writing markers (%WC) are provided to control and monitor this interface:

9.1 STATUS READING WORDS

CANopen Master and Slave Status: set of reading markers to provide information about the general status of the CANopen master and the communication status between the master and each of the slaves.	
Marker	**Description**
%RS4000	CANopen master status: Bit 0: all slaves were contacted. Bit 1: download of the slave configurations was performed. Bit 2: slave error control started. Bit 3: end of the initialization of the slaves. Bit 4: error detected in the initialization of at least one slave. Bit 5: error detected in the error control task of at least one slave. Bits 6 and 7: reserved Bit 8: it takes on the toggle bit value (see %CD3200) after the master sends NMT command. Bits 9 ... 12: reserved Bit 13: CAN interface in the bus off status. Bit 14: no power supply on the CAN interface. Bit 15: communication disabled
%RS4001 ... %RS4127	CANopen slave status. They are 127 Word markers, seeing that each marker is linked to an address on the CANopen network, and indicates the slave status at the address: Bit 0: master contacted slave successfully. Bit 1: download of the master configurations was performed successfully. Bit 2: slave error control started. Bit 3: end of the initialization of the slave. Bit 4: error detected at the initialization of the slave. Bit 5: error detected in the slave error control task. Bits 6 ... 15: reserved

Last Error at the SDO Client: set of reading markers to report data about errors at the SDO client. If a request is made to the SDO client and the slave does not respond, or responds with an error, the data related to the last error detected by the SDO client are saved on these markers.	
Marker	**Description**
%RS4128	Address of the destination slave to which the SDO request was sent.
%RS4129	Index of the object accessed via SDO.
%RS4130	Sub-index of the objected accessed.
%RS4131	Type of access performed: 1 = reading, 2 = writing.
%RS4132 ... %RS4133	For writing access, it indicates the written value.
%RS4134 ... %RS4135	It indicates the code of the error received, according to the communication errors via SDO of the CANopen protocol specification.

Last EMCY detected: set of reading markers to report data about errors informed by EMCY producers. The CANopen master does not have EMCY consumer. EMCY telegrams sent by network slaves, however, are captured by the master, and the information of the last EMCY detected is saved on these markers.	
Marker	**Description**
%RS4136	Address of the slave which reported the EMCY.
%RS4137... % RS4140	Eight bytes of data of the EMCY telegram, with information about the error code informed by the slave.

9.2 COMMAND WRITING WORDS

CANopen Master Control: set of writing markers to control the CANopen master.	
Marker	**Description**
%WC4142	Command to control the CANopen master and send NMT telegram. Bits 0 ... 7: NMT command code: 1 = START 2 = STOP 128 = ENTER PRE-OPERATIONAL 129 = RESET NODE 130 = RESET COMMUNICATION Bit 8: toggle bit, whenever the value of this bit is changed, it sends the programmed command. Bits 9 ... 14: reserved Bit 15: disable CANopen communication
%WC4143	Bits 16 ... 23: destination slave address for sending the NMT command.

10 FAULTS AND ALARMS RELATED TO THE CANOPEN COMMUNICATION

A133/F33 – CAN INTERFACE WITHOUT POWER SUPPLY

Description:
It indicates that the CAN interface does not have power supply between the pins 1 and 5 of the connector.

Actuation:
In order that it be possible to send and receive telegrams through the CAN interface, it is necessary to supply external power to the interface circuit.

If the CAN interface is connected to the power supply and the absence of power is detected, the alarm A133 – or the fault F33, depending on the P0662 programming, will be signalized through the HMI. If the circuit power supply is reestablished, the CAN communication will be reinitiated. In case of alarms, the alarm indication will also be removed from the HMI.

Possible Causes/Correction:
- Measure the voltage between the pins 1 and 5 of the CAN interface connector.
- Verify if the power supply cables have not been changed or inverted.
- Make sure there is no contact problem in the cable or in the CAN interface connector.

A134/F34 – BUS OFF

Description:
The *bus off* error in the CAN interface has been detected.

Actuation:
If the number of reception or transmission errors detected by the CAN interface is too high[9], the CAN controller can be taken to the *bus off* state, where it interrupts the communication and disables the CAN interface.

In this case the alarm A134 – or the fault F34, depending on the P0662 programming, will be signalized through the HMI. In order that the communication be reestablished, it will be necessary to cycle the power of the product, or remove the power supply from the CAN interface and apply it again, so that the communication be reinitiated.

Possible Causes/Correction:
- Verify if there is any short-circuit between the CAN circuit transmission cables.
- Verify if the cables have not been changed or inverted.
- Verify if all the network devices use the same baud rate.
- Verify if termination resistors with the correct values were installed only at the extremes of the main bus.
- Verify if the CAN network installation was carried out in proper manner.

A135/F35 – NODE GUARDING/HEARTBEAT

Description:
The CANopen communication error control detected a communication error by using the guarding mechanism.

Operation:
By using the error control mechanisms – Node Guarding or Heartbeat – the master and the slave can exchange periodic telegrams, with a predetermined period. If the communication is interrupted by some reason, the master, as well as the slave, will be able to detect communication error through the timeout in the exchange of those messages.

[9] For more information on the error detection, refer to the CAN specification.

In this case the alarm A135 or the fault F35, depending on the P0662 programming, will be signalized through the HMI. In case of alarms, the alarm indication will be removed from the HMI if this error control is enabled again.

Possible Causes/Correction:

- Verify the times programmed in both master and slave, for the message exchanging. In order to avoid problems due to transmission delays and differences in the time counting, it is recommended that the values programmed for message exchanging in the master be a little bit shorter than the times programmed for the error detection by the slave.
- Verify if the master is sending the *guarding* telegrams in the programmed time.
- Verify communication problems that can cause telegram losses or transmission delays.

WEG Equipamentos Elétricos S.A.
Jaraguá do Sul – SC – Brasil
Fone 55 (47) 3276-4000 – Fax 55 (47) 3276-4020
São Paulo – SP – Brasil
Fone 55 (11) 5053-2300 – Fax 55 (11) 5052-4212
automacao@weg.net
www.weg.net