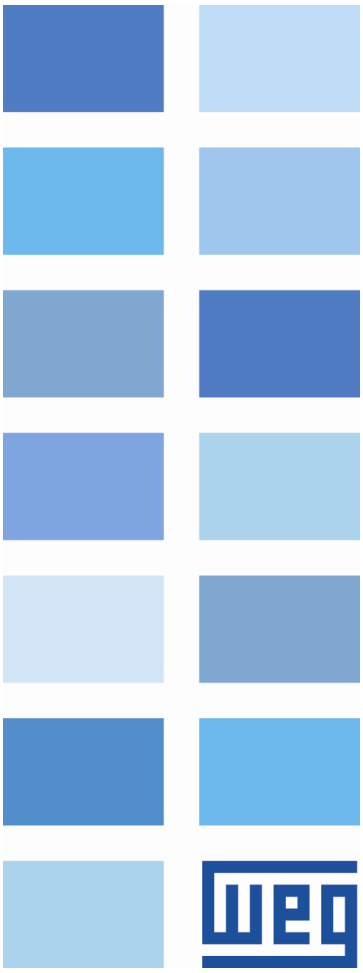
CANopen

SCW100

User's Guide



CANopen User's Guide

Series: SCW100

Language: English

Document: 10008601670 / 01

Build 221

Publication Date: 04/2022

The information below describes the reviews made in this manual.

Version	Revision	Description
-	R00	Cancelled
-	R01	First edition

CONTENTS

ABOUT THE MANUAL

This manual supplies the necessary information for the operation of the SCW100 starter manager using the CANopen protocol. This manual must be used together with the SCW100 user's manual .

ABBREVIATIONS AND DEFINITIONS

ASCII American Standard Code for Information Interchange
CAN Controller Area Network
CiA CAN in Automation
CIP Common Industrial Protocol
CRC Cycling Redundancy Check
HMI Human-Machine Interface
ISO International Organization for Standardization
ODVA Open DeviceNet Vendor Association
OSI Open Systems Interconnection
PLC Programmable Logic Controller
ro Read only
rw Read/write
RTR Remote Transmission Request

NUMERICAL REPRESENTATION

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

DOCUMENTS

The CANopen protocol 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 303-3 CANopen Indicator Specification	1.0	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
Planning and Installation Manual - DeviceNet Cable System	PUB00027R1	ODVA

1 MAIN CHARACTERISTICS

Below are the main characteristics for communication of the starter manager SCW100 with CANopen accessory.

- Network management task (NMT).
- 5 transmission PDOs.
- 2 reception PDOs.
- Heartbeat Consumer.
- Heartbeat Producer.
- Node Guarding.
- SDO Client.
- SYNC producer/consumer.
- It is supplied with an EDS file for the network master configuration.
- Acyclic data available for parameterization.

2 INTERFACE DESCRIPTION

2.1 CHARACTERISTICS OF THE CAN INTERFACE

- Interface galvanically insulated and with differential signal, providing more robustness against electromagnetic interference.
- External power supply of 24 V.
- It allows the device to operate as CANopen slave.
- Allows data communication for equipment operation and parameterization.
- Enables communication using baud rates from 20 Kbits/s up to 1 Mbit/s.
- Maximum bus length of 1000 meters.

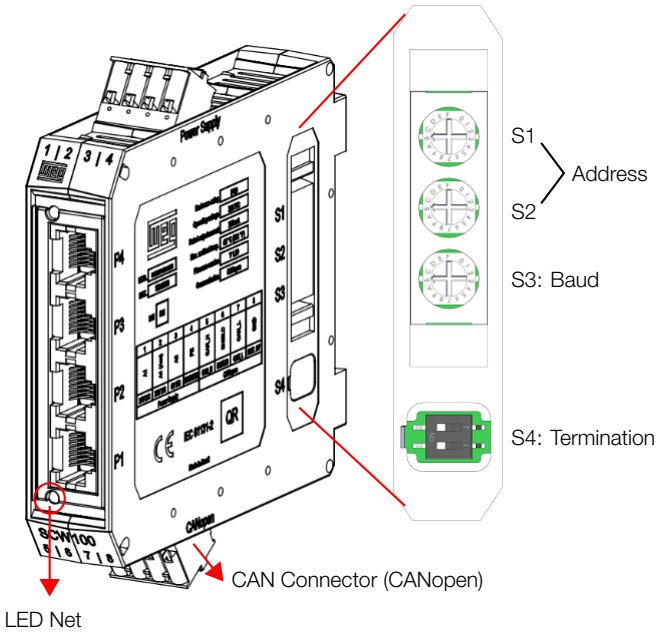


Figure 2.1: *Connector, indications and configurations for SCW100*

2.2 CONNECTOR

The interface is available through a 4-wire plug-in connector with the following pin assignment:

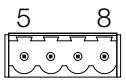


Table 2.1: *Pin assignment of connector for CANopen interface*

Pin	Name	Function
5	CAN_H	Communication signal CAN_H
6	Shield	Cable shield
7	CAN_L	Communication signal CAN_L
8	CAN_GND	0V for CAN insulated interface circuit

2.3 ADDRESS

The product address is configured using two hex keys S1 and S2 as illustrated in the figure 2.2.

- Valid addresses: 1 to 127 (01h to 7Fh);

NOTE!
For the changes in this setting be effective, the equipment must be powered off and on again.

The address selected using the keys represents a hexadecimal value. Therefore, it must be converted to decimal, when necessary, in the parameterization of the CANopen master.

The swtiches to configure the CANopen slave address are illustrated in more detail in the figure 2.2, where switch S1 is the least significant and switch S2 is the most significant:

- Switch S1 = Eh
- Switch S2 = 2h

Since switch S1 is the least significant, the address composed by the switches is 2Eh, or 46 decimal.

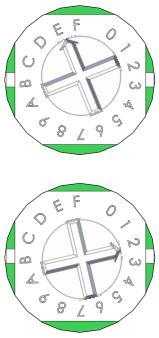


Figure 2.2: *Rotary switches for address configuration*

2.4 BAUD RATE

The SCW100 starter manager has a rotary S3 hex switch that allows you to configure the communication rate. According to the position of the switch, this configuration is programmed as indicated in the table 2.2.

Table 2.2: *S3 switch settings for programming the baud rate*

Switch position	Baud rate
0	1 Mbit/s
1	800 Kbit/s
2	500 Kbit/s
3	250 Kbit/s
4	125 Kbit/s
5	100 Kbit/s
6	50 Kbit/s
7	20 Kbit/s

NOTE!
For the changes in this setting be effective, the equipment must be powered off and on again.

2.5 TERMINATING RESISTOR

The product has a switch (S4) that can be activated to enable the termination resistor according to figure 2.1. The configurations of the switch to enable the termination resistor are shown in table 2.3.

Table 2.3: *Configurations of the switch (S4) that enables the termination resistor*

Switch Setting	Option
SW.1 = OFF and SW.2 = OFF	CAN termination off
SW.1 = ON and SW.2 = ON	CAN termination on
SW.1 = OFF and SW.2 = ON	Combination not allowed
SW.1 = ON and SW.2 = OFF	

2.6 INDICATION LED

The SCW100 starter manager has a bicolor LED (green and red), as shown in figure 2.1, the tables below show the behavior of these LEDs depending on the status of the starter manager:

Table 2.4: *LED NET - GREEN*

Indication	State	Description
Off	-	Powered off.
Green, one flash	Stopped	Device is in the "Stopped" state. PDOs and SDOs are not available at this state.
Green, alternating each 200ms	Pre-operational	Device is in the "Pre-operational" state. PDOs are not available for communication.
Green, solid	Operational	Device is in the "Operational" state.

Table 2.5: *LED NET - RED*

Indication	State	Description
Off	No error	Device is operating normally.
Red, one flash	Warning or Passive	Indicates the CAN interface is in Warning or Error Passive state. It may occur, for instance, it is the only device connected to network.
Red, two flashes	Node Guarding or Heartbeat error	An comminication error has been detected using CANopen error control mechanism - guarding or heartbeat.
Red, solid	BUS OFF error	CAN interface is in the bus-off state. It indicates a critical communication error condition, normally associated with installation problems or incorrect baud rate configuration. It is necessary to power cycle the device to restore communication.
Red, alternating each 50ms	CANopen not initialized	CANopen communication could not be properly initialized. Verify if it is configured to a valid address (01h – 7Fh).

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.

NOTE!
Detailed recommendations on how to perform the installation are available at document "Planning and Installation Manual" (item DOCUMENTS).

3.1 BAUD RATE

Equipments with CANopen interface generally allow the configuration of the desired baud rate, ranging from 10 kbit/s to 1 Mbit/s. The baud rate that can be used by the equipment depends on the length of the cable used in the installation. The table 3.1 shows the baud rates and the maximum cable length that can be used in the installation, according to the protocol recommendation.

Table 3.1: *Supported baud rates and cable length*

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

All network equipment must be programmed to use the same communication baud rate.

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.

3.3 TERMINATION RESISTOR

The use of termination resistors at the ends of the bus is essential to avoid line reflection, which can impair the signal and cause communication errors. Termination resistors of 121 Ω | 0.25 W must be connected between the signals CAN_H and CAN_L at the ends of the main bus.

3.4 CABLE

The connection of CAN_L and CAN_H signals must be 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)	Resistence per Meter (mΩ/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.

3.5 CONNECTION TO 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.

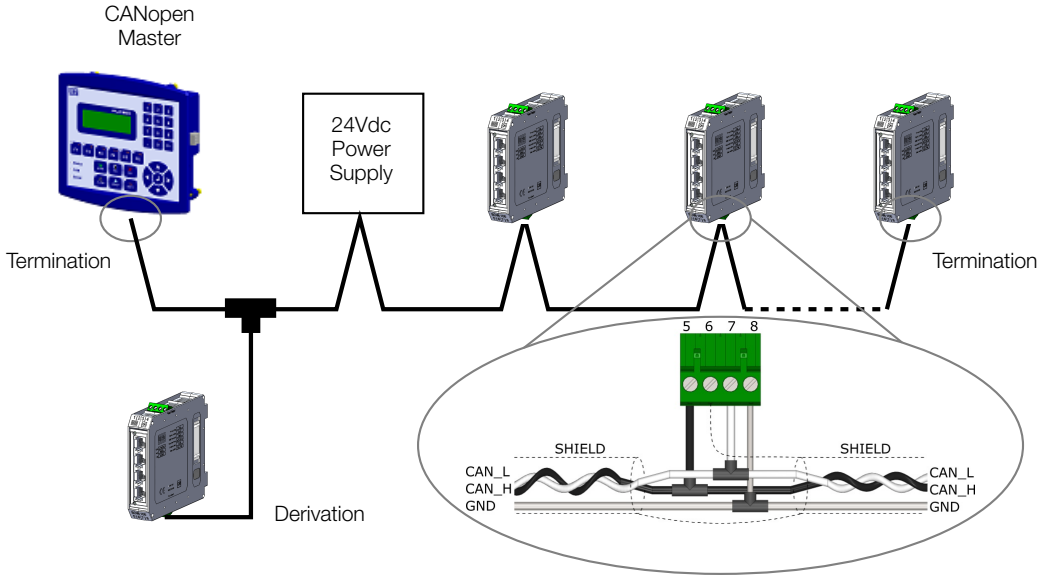


Figure 3.1: CANopen network installation example

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.

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. Use the power supply to power the bus cable system only.

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 S STATUS

Allows accessing starter manager SCW100 reading variables.

S4 COMMUNICATION

Read parameters that display information about the communication interfaces for the product.

S4 Communication
P0030: CAN - Address
Range: 1 ... 127 **Default:** 1

Description:
It allows viewing the address configured for CAN communication, programmed using DIP switches S1 and S2.

S4 Communication
P0031: CAN - Baudrate
Range: 0 ... 7 **Default:** 0

Description:
It allows viewing the value of the baud rate for the CAN interface, programmed using DIP switch S3, in bits per second.

Indication	Description
0 = 1 Mbit/s	CAN baud rate.
1 = 800 Kbit/s	CAN baud rate.
2 = 500 Kbit/s	CAN baud rate.
3 = 250 Kbit/s	CAN baud rate.
4 = 125 Kbit/s	CAN baud rate.
5 = 100 Kbit/s	CAN baud rate.
6 = 50 Kbit/s	CAN baud rate.
7 = 20 Kbit/s	CAN baud rate.

S4 Communication
P0033: CAN - Controller Status
Range: 0 ... 5 **Default:** 0

Description:
It allows identifying if the CAN interface is enabled and if the communication presents errors.

Indication	Description
0 = Disabled	Inactive CAN interface.
1 = Reserved	-
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.

S4 Communication
P0034: CAN - RX CAN Telegrams
Range: 0 ... 65535 **Default:** 0

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.

S4 Communication

P0035: CAN - TX CAN Telegrams

Range: 0 ... 65535 **Default:** 0

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.

S4 Communication

P0036: CAN - Bus Off counter

Range: 0 ... 65535 **Default:** 0

Description:
It is a cyclic counter that indicates the number of times the device entered the bus off state in the CAN network.

S4 Communication

P0037: CAN - Lost Telegrams

Range: 0 ... 65535 **Default:** 0

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.

S4 Communication

P0038: CAN - CANopen Communication Status

Range: 0 ... 5 **Default:** 0

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

Indication	Description
0 = Disabled	CANopen protocol disabled.
1 = Reserved	-
2 = Comm Enabled	Communication enabled.
3 = Error Ctrl. Enab.	Communication enabled and error control service enabled (Node Guarding/Heartbeat).
4 = Guarding Error	Node Guarding error occurred.
5 = Heartbeat Error	Heartbeat error occurred.

S4 Communication

P0039: CAN - CANopen Slave Status

Range: 0 ... 4 **Default:** 0

Description:
Each slave in the CANopen network has a state machine that controls its behavior regarding the communication. This parameter indicates in which state the device is.

Indication	Description
0 = Disabled	CANopen protocol disabled.
1 = Initialization	Communication with the device is not possible during this stage, which is concluded automatically.
2 = Stopped	Only the NMT object is available.
3 = Operational	All the communication objects are available.
4 = PreOperational	It is already possible to communicate with the slave but its PDOs are not yet available for operation.

5 C CONFIGURATION

Allows accessing starter manager SCW100 writing variables.

C2 COMMUNICATION

Write parameters that allow configure the communication interfaces for the product.

C2 Communication
P0101: Action for Communication Fault
Range: 0 ... 1 **Default:** 1

Description:
It allows configuring the protection tripping mode for communication error.

Indication	Description
0 = No Action	No tripping.
1 = Disable Outputs	Turns off all product outputs. It is equivalent to receiving the stop command for the starts, and clearing the command word for the digital outputs.

C2 Communication
P0032: CAN - Bus Off Reset
Range: 0 ... 1 **Default:** 0

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

Indication	Description
0 = Manual	If bus off occurs, LED will indicate this condition and the communication will be disabled. The action programmed in parameter P0101 - Action for Communication Fault will be performed. In order that the device communicates again through the CAN interface, it will be necessary to power cycle the device.
1 = Automatic	If bus off occurs, the communication will be reinitiated automatically and the error will be ignored. In this case it will not indicate this error at LEDs and the device will not perform any error action.

6 OPERATION IN THE CANOPEN NETWORK

6.1 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.

6.2 CYCLIC DATA

Cyclic data is the data normally used for status monitoring and equipment control. For CANopen protocol, the interface supports 2 receive PDOs and 5 transmit PDOs.

It is necessary the configuration to be made both at the slave and master.

6.3 ACYCLIC DATA

In addition to the cyclic data, the interface also provides acyclic data via SDO. Using this type of communication, you can access any equipment parameter. Access to this type of data is commonly done using instructions for reading or writing data, which should indicate the index and sub-index to the desired parameter. The item 7.4 describes how to address the parameters for SCW100 starter manager.

6.4 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 following communication objects (COBs) are described by the specification:

Table 6.1: *Types of Communication Objects (COBs)*

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.).
Syncronization 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: device control service, with which the master controls the state of each network slave, and error control service (Node Guarding an Heartbeat), with which the device sends periodic messages to inform that the connection is active.

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

6.5 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.
- Adrress: indicates with which network device the telegram is linked.

A table with the standard values for the different communication objects 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 6.2: *COB-ID for the different objects*

COB	Function Code (bits 10-7)	COB-ID 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)

6.6 EDS FILE

Each device on an CANopen network has an EDS configuration file, which contains information about the device functions on the network. This file is used by a master or configuration software to program devices present at CANopen network.

The EDS file is available from WEG website (http://www.weg.net). It is important to note if the EDS configuration file is compatible with the firmware version of the SCW100 starter manager.

7 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 7.1: *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.

7.1 DICTIONARY STRUCTRE

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.

7.2 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.

7.3 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 starter manager SCW100 is presented next.

Table 7.2: *Lista de objetos – 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	consume 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
Receive PDO Mapping Parameter				
1600h	RECORD	1st receive PDO mapping	PDO Mapping	rw
1601h	RECORD	2nd 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
1802h	RECORD	3rd transmit PDO Parameter	PDO CommPar	rw
1803h	RECORD	4th transmit PDO Parameter	PDO CommPar	rw
1804h	RECORD	5th 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
1A02h	RECORD	3rd transmit PDO mapping	PDO Mapping	rw
1A03h	RECORD	4th transmit PDO mapping	PDO Mapping	rw
1A04h	RECORD	5th transmit PDO mapping	PDO Mapping	rw

These objects can only be read and written via the CANopen network, it is not available in 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 8 for more details on the available objects in this range of the objects dictionary.

7.4 MANUFACTURER 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 SCW100, the whole list of parameters was made available in this object range. It is possible to operate the SCW100 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 Net Id to this index their position in the dictionary is obtained. To identify how the parameters are distributed in the object dictionary, refer to the item A.

In order to be able to program the SCW100 operation correctly via the CANopen network, it is necessary to know its operation through the parameters.

Refer to the SCW100 starter manager programming manual for a complete list of the parameters and their detailed description.

8 COMMUNICATION OBJECTS DESCRIPTION

This item describes in detail each of the communication objects available for the SCW100 starter manager. It is necessary to know how to operate these objects to be able to use the available functions for the SCW100 starter manager communication.

8.1 IDENTIFICATION OBJECT

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.

8.1.1 Object 1000h - Device Type

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

Table 8.1: *Object 1000h - Device Type*

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1000h	0	Device Type	UNSIGNED32	RO	No	0

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.

8.1.2 Object 1001h - Error Register

This object indicates whether or not an error in the device occurred. The type of error registered for the equipment follows what is described in the table 8.2.

Table 8.2: *Object 1001h - Error Register*

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1001h	0	Error register	UNSIGNED8	RO	yes	0

Table 8.3: *Structure of the object Error Register*

Bit	Meaning
0	Generic error
1	Currrent
2	Voltage
3	Temperature
4	Communication
5	Reservaded (always 0)
6	Reservaded (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.

8.1.3 Object 1018h - Identity Object

It brings general information about the device.

Table 8.4: *Object 1018h - Identity Object*

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1018h	0	Number of the last sub-index	UNSIGNED8	RO	No	4
	1	Vendor ID	UNSIGNED32	RO	No	0000.0123h
	2	Product code	UNSIGNED32	RO	No	0000.1200h
	3	Revision number	UNSIGNED32	RO	No	According to the equipment firmware version
	4	Serial number	UNSIGNED32	RO	No	Different for each SCW100

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 starter manager SCW100 in CANopen network.

8.2 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.

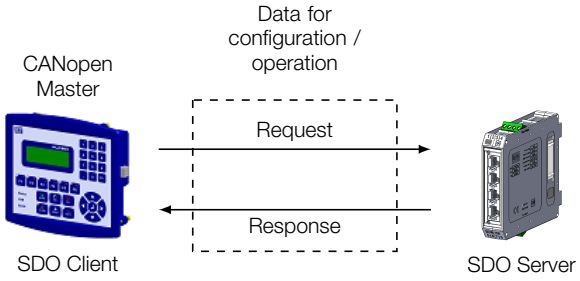


Figure 8.1: *Communication between SDO client and server*

8.2.1 Object 1200h - SDO Server

The starter manager SCW100 starter manager 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 the SCW100 operation. Every SDO server has an object, of the SDO_PARAMETER type, for its configuration, having the following structure:

Table 8.5: *Objet 1200h - SDO Server*

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1200h	0	Number of the last sub-index	UNSIGNED8	RO	No	2
	1	COB-ID Client - Server (rx)	UNSIGNED32	RO	No	600h + Node-ID
	2	COB-ID Server - Client (tx)	UNSIGNED32	RO	No	580h + Node-ID

8.2.2 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		Subindex	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 8.6: *Command codes for SDO client*

Command	Function	Description	Object Data
22h	Download	Write object	Not defined
23h	Download	Write object	4 byte
2Bh	Download	Write object	2 byte
2Fh	Download	Write object	1 byte
40h	Upload	Read object	Not used
60h ou 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 one 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 8.7: *Command codes for SDO server*

Command	Function	Description	Object Data
60h	Download	Write object	Not used
43h	Upload	Write object	4 byte
4Bh	Upload	Write object	2 byte
4Fh	Upload	Write object	1 byte
41h	Upload segment	Initiates segmented response for read	4 byte
01h ou 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 SCW100 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 slave 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		Subindex	Data			
601h	40h	00h	20h	00h	00h	00h	00h	00h

The slave responds to the request indicating that the value of the referred object is equal to 999 [1]:

Identifier	Command	Index		Subindex	Data			
581h	4Bh	00h	20h	00h	E7h	03h	00h	00h

8.3 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 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.

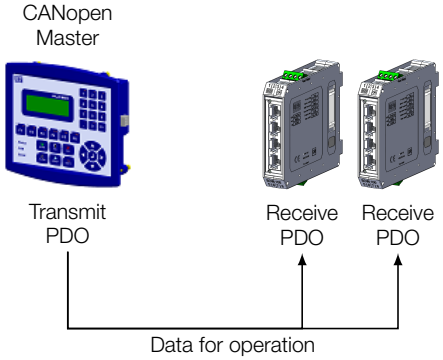


Figure 8.2: *Communication using PDOs*

NOTE!
PDOs can only be transmitted or received when the device is in the operational state.

8.3.1 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 the manufacturer’s specific objects (2000h – 5FFFh), the table A 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.

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

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

8.3.2 Receive PDOs

The receive PDOs, or RPDOs, are responsible for receiving data that other devices send to the CANopen network. The starter manager SCW100 has 2 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	Sub-index	Name	Type	Access	PDO Mapping	Value
1400h - 1401h	0	Number of the last sub-index	UNSIGNED8	RO	No	2
	1	COB-ID used by the PDO	UNSIGNED32	RW	No	200h / 300h + Node-ID
	2	Transmission Type	UNSIGNED8	RW	No	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 8.8: *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 SCW100 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 8.9: *Transmission type description*

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 e 253:** not allowed for receive PDOs.
- **Values 254 e 255:** they indicated that there is no relationship with the synchronization object. When receiving a message, its values are updated immediately.

PDO_MAPPING

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1600h - 1601h	0	Number of mapped objects	0 = disable 1-4=number of mapped objects	RO	No	0
	1 - 4	1 up to 4 object mapped in the PDO	UNSIGNED32	RW	No	According EDS file

This parameter indicates the mapped objects in the SCW100 starter manager receive PDOs. The default value of these objects is indicated in the product's EDS file. 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[2] and size (in bits) in an UNSIGNED32, field with the following format:

UNSIGNED32		
Index (16 bits)	Sub-index (8 bits)	Objects size (8 bits)

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

- **Sub-index 0 = 3:** This PDO has three mapped objects.
- **Sub-index 1 = 244C.0010h:** the first mapped object has an index equal to 244Ch, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P1100 - Forward Start Command.
- **Sub-index 2 = 2451.0010h:** the second mapped object has an index equal to 2451h, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P1105 - Reverse Start Command.
- **Sub-index 3 = 2456.0010h:** the second mapped object has an index equal to 2456h, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P1110 - Stop Command.

It is possible to modify this mapping by changing the quantity or the number of mapped objects. Remembering that only 4 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 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.
- Do not forget that PDOs can only be received if the device is in the operational state.

8.3.3 Transmit PDOs

The transmit PDOs, or TPDOs, as the name says, are responsible for transmitting data for the CANopen network. The starter manager SCW100 starter manager has 5 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

[2] 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	Sub-index	Name	Type	Access	PDO Mapping	Value
1800h - 1804h	0	Number of the last sub-index	UNSIGNED8	RO	No	5
	1	COB-ID used by the PDO	UNSIGNED32	RW	No	180h / 280h / 380h / 480h / 580h + Node-ID
	2	Transmission Type	UNSIGNED8	RW	No	254
	3	Time between transmissions	UNSIGNED16	RW	No	-
	4	Compability entry	UNSIGNED8	RW	No	-
	5	Event timer	UNSIGNED16	RW	No	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 8.8.

The sub-index 2 indicates the transmission type of this object, which follows the table 8.9 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.
- **Value 254:** The object must be transmitted according to the timer programmed in sub-index 5.
- **Value 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 100 µs) 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 4 has no function and exists only for compatibility reasons.

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 1 ms), 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 starter manager SCW100 is 2 ms.
- 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).
- Do not forget that PDOs can only be received if the slave is in the operational state.

PDO_MAPPING

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1A00h - 1A04h	0	Number of the last sub-index	UNSIGNED8	RO	No	0
	1 - 4	1 up to 4 object mapped in the PDO	UNSIGNED32	RW	No	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)	Object size (8 bits)

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

- **Sub-índice 0 = 4:** This TPDO has four mapped objects.
- **Sub-índice 1 = 20C8.0010h:** the first mapped object has an index equal to 20C8h, sub-index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P0200 - P1 Status.
- **Sub-índice 2 = 20C9.0010h:** the second mapped object has an index equal to 20C9h, sub- index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P0201 - P1 Status - Contactor.
- **Sub-índice 3 = 20CA.0010h:** the second mapped object has an index equal to 20CAh, sub- index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P0202 - P1 Status - Error.
- **Sub-índice 4 = 20CB.0020h:** the second mapped object has an index equal to 20CBh, sub- index 0 (zero), and a size of 16 bits. This object corresponds to the parameter P0203 - P1 Status - Alarm.

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.

8.4 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 starter manager SCW100 has the function of a SYNC consumer and, therefore, it can program its PDOs to be synchronous. Synchronous PDOs are those related to the synchronization object, thus they can be programmed to be transmitted or updated based in this object.

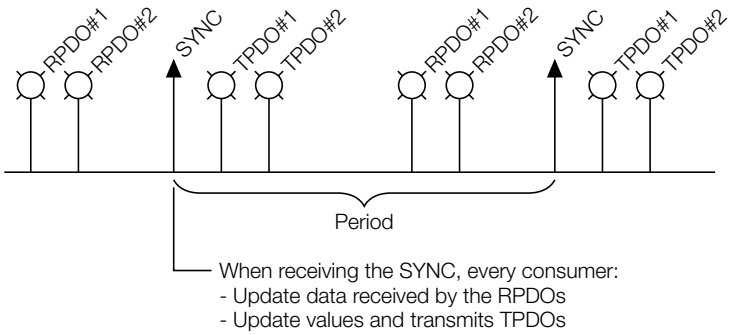


Figure 8.3: *SYNC*

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. The following object is available for the configuration of the SYNC consumer:

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1005h	0	COB-ID SYNC	UNSIGNED32	RW	No	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.

8.5 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 this object, the services of node control and error control are available (using Node Guarding or Heartbeat).

8.5.1 Slave State Control

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

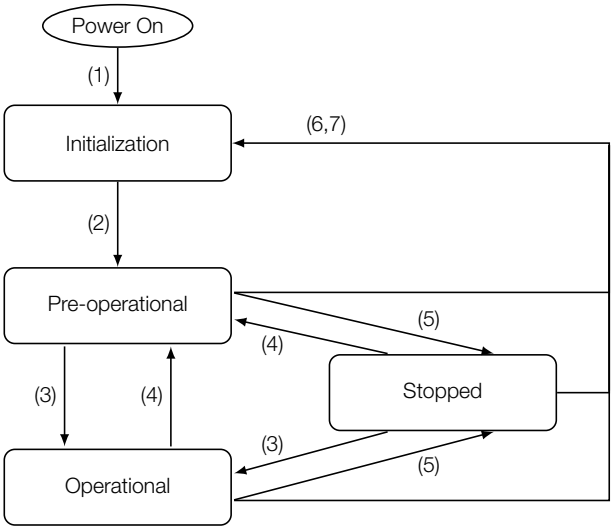


Figure 8.4: *CANopen node state diagram*

Table 8.10: *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 8.11: *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 8.12: *Commands for the state transition*

Command Code	Destination Node ID
1 = START node (transition 3) 2 = STOP node (transition 4) 128 = Enter preoperational (transition 5) 129 = Reset node (transition 6) 130 = Reset comunication (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 8.4). The Reset node command makes the slave 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.

8.5.2 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.

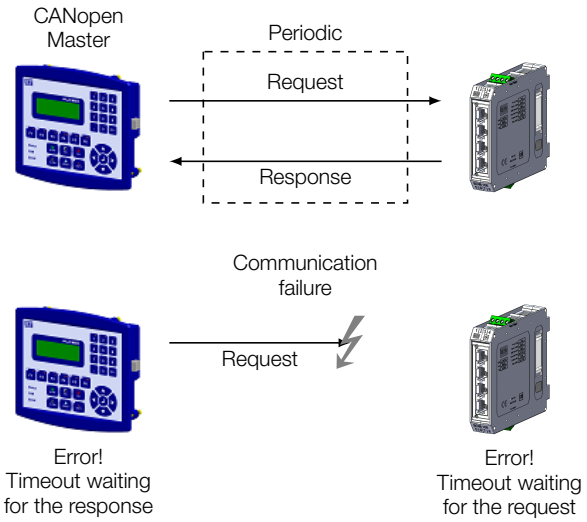


Figure 8.5: *Error control service – Node Guarding*

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

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
100Ch	0	Guard Time	UNSIGNED16	RW	No	0

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
100Dh	0	Life Time Factor	UNSIGNED8	RW	No	0

The 100Ch object allows programming the time necessary (in milliseconds) for a fault occurrence being detected, in case the slave 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 slave starts counting these times starting from the first Node Guarding telegram received from the network master. The master's 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:

Identificador	byte 1	
	bit 7	bit 6 ... 0
700h + Node ID	Toogle	Estado do Escravo

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 starter manager SCW100 detects an error using this mechanism, it will turn automatically to the preoperational state and indicate with the LED of error.

NOTE!

- This object is active even in the stopped state (see table 8.11).
- 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.
- The minimum value accepted by the SCW100 starter manager is 2 ms. 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 slave only one of the two services - Heartbeat or Node Guarding – can be enabled.

8.5.3 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.

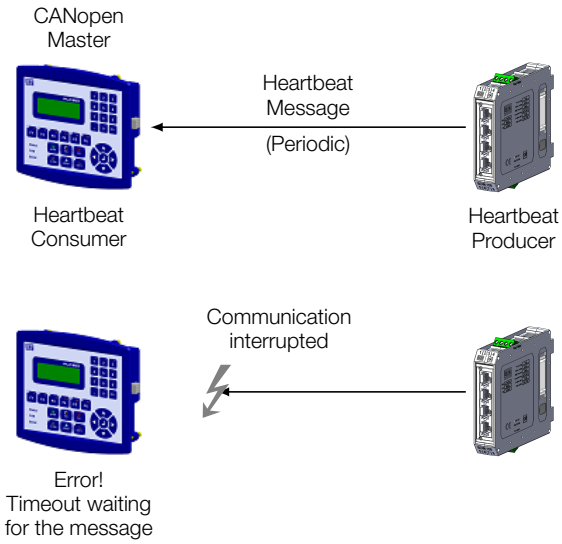


Figure 8.6: *Error control service – Heartbeat*

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 SCW100 starter manager has the producer and consumer of heartbeat services. As a consumer, it is possible to program different producers to be monitored by the equipment:

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1016h	0	Number of the last sub-index	UNSIGNED8	RO	No	4
	1 to 4	Consumer Heartbeat Time 1 to 4	UNSIGNED32	RW	No	0

At sub-indexes 1 to 4, 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, it will turn automatically to the preoperational state and indicate with the LED of error.

As a producer, the SCW100 starter manager has an object for the configuration of that service:

Index	Sub-index	Name	Type	Access	PDO Mapping	Value
1017h	0	Producer Heartbeat Time	UNSIGNED16	RW	No	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 device initiates the transmission of messages with the following format:

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

NOTE!

- This object is active even in the stopped state (see table 8.11).
- 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.
- The minimum value accepted by the SCW100 starter manager is 2 ms. 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 slave only one of the two services - Heartbeat or Node Guarding – can be enabled.

8.6 INITIALIZATION PROCEDURE

Once the operation of the objects available for the SCW100 starter manager 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:

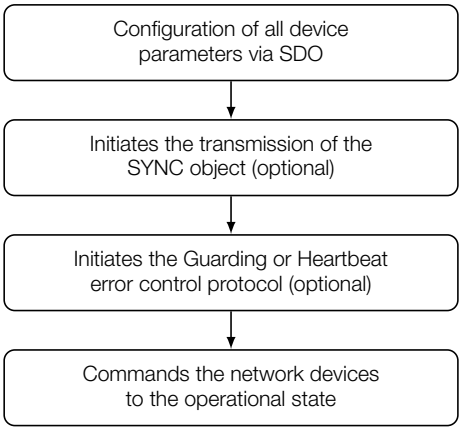


Figure 8.7: *Initialization process flowchart*

It is necessary to observe that the SCW100 starter manager 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.

9 STARTUP GUIDE

The main steps to start up the SCW100 starter manager in CANopen network are described below. These steps represent an example of use. Check out the specific chapters for details on the indicated steps.

9.1 INSTALLING THE PRODUCT ON THE NETWORK

1. Install SCW100 starter manager on the CANopen network and make the necessary configurations for operation, as described in item 2.
2. Connect the cable to the accessory, considering the recommended instructions in network installation, as described in item 3.5:
   - Use shielded cable.
   - Properly ground network equipment.
   - Avoid laying communication cables next to power cables.

9.2 CONFIGURING THE EQUIPMENT

1. Follow the recommendations described in the user manual to program the device parameters.
2. Configure communication setting, such as address and baud rate at DIP switch S2.
3. Program the desired action for the equipment in case of communication fault in P0101: Action for Communication Fault.

9.3 CONFIGURING THE MASTER

The way the network configuration is done depends greatly on the used client and the configuration tool. It is essential to know the tools used to perform this activity. In general, the following steps are necessary to perform the network configuration.

1. Load the EDS file[3] to the list of devices in the network configuration tool.
2. Select SCW100 starter manager from the available list of devices on the network configuration tool. This can be done manually or automatically, if allowed by the tool.
3. During the configuration of the network, it is necessary to define which data will be read and written at starter manager SCW100 by configuring the transmission and reception PDOs as described in item 8.3. Among the main parameters that can be used to control the device, we can mention:
4. Configure error control using the Node Guarding or Heartbeat services as described in item 8.5.

Once configured, the network status P0038: CAN - CANopen Communication Status indicates 2 or 3 and the node state P0039: CAN - CANopen Slave Status indicates 3. It is in this condition that PDO transmission and reception effectively occurs.

9.4 COMMUNICATION STATUS

Once the network is assembled and the client programmed, it is possible to use the Net LED and parameters of the equipment to identify some status related to the communication.

[3]The EDS file is available from WEG website (http://www.weg.net). It is important to note if the EDS configuration file is compatible with the firmware version of the SCW100 starter manager.

- The Net LED provide information about the status of the interface.
- The parameters P0038: CAN - CANopen Communication Status and P0039: CAN - CANopen Slave Status indicate the status of CANopen communication.

The master of the network must also supply information about the communication with the slave.

9.5 OPERATION USING PROCESS DATA

Once the communication is established, the data mapped in the PDOs is automatically updated.

It is important to know these parameters to program the master as desired for the application.

9.6 ACCESS TO PARAMETERS – ACYCLIC MESSAGES

Besides the cyclic communication using PDOs, the CANopen protocol also defines a kind of acyclic message via SDO, used especially in asynchronous tasks, such as parameter setting and configuration of the equipment.

The EDS file provides the full parameter list of the equipment, which can be accessed via SDO. The item 7.4 describes how to address the parameters of the starter manager SCW100 via SDO.

10 FAULTS AND ALARMS

Fault/Alarm	Description	Possible Causes
Bus Off	The bus off error in the CAN interface has been detected. If the number of reception or transmission errors detected by the CAN interface is too high, the CAN controller can be taken to the bus off state, where it interrupts the communication and disables the CAN interface. 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. In this case, it will be signaled through the Net LED on the starter manager. Communication is reestablished automatically if the P0032 - CAN - Bus Off Reset parameter has a value of 1. If the value of the P0032 - CAN - Bus Off Reset parameter is 0, it will be necessary to turn the remote unit off and on to reestablish communication.	- 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.
CANopen Offline	It occurs when CANopen node state changes from operational to pre-operational.	- Verify the error control mechanisms operation (Heartbeat/Node Guarding). - Verify if the master is sending the guarding/heartbeat telegrams in the programmed time. - Verify communication problems that can cause telegram losses or transmission delays.

APPENDIX A QUICK REFERENCES

Table A.2: *Parameters quick reference*

Parameter	Description	Range of values	Factory setting	Properties	Communication Address	CANopen Index	Sub-Index
S1 Status\Product Information							
P0001	Firmware Version	0.0 to 655.35	-	ro, 16bit	1	2001h	0
P0010	Rotary Switch S1	0 to 15	-	ro, 8bit	10	200Ah	0
P0011	Rotary Switch S2	0 to 15	-	ro, 8bit	11	200Bh	0
P0012	Rotary Switch S3	0 to 15	-	ro, 8bit	12	200Ch	0
P0015	Status of the Digital Inputs	Bit 0 = DI1 Bit 1 = DI2 Bit 2 = DI3 Bit 3 = DI4 Bit 4 = DI5 Bit 5 = DI6 Bit 6 = DI7 Bit 7 = DI8 Bit 8 = DI9 Bit 9 = DI10 Bit 10 = DI11 Bit 11 = DI12	-	ro, 16bit	15	200Fh	0
P0016	Status of the Digital Outputs	Bit 0 = DO1 Bit 1 = DO2 Bit 2 = DO3 Bit 3 = DO4 Bit 4 = DO5 Bit 5 = DO6 Bit 6 = DO7 Bit 7 = DO8	-	ro, 16bit	16	2010h	0
P0500	CPU Temperature	-100 to 100 °C	-	ro, s16bit	500	21F4h	0
S2 Status\Starts							
P0120	P1 Contactor 1 Closing Time	0 to 65535 ms	-	ro, 16bit	120	2078h	0
P0121	P1 Contactor 1 Opening Time	0 to 65535 ms	-	ro, 16bit	121	2079h	0
P0122	P1 Contactor 2 Closing Time	0 to 65535 ms	-	ro, 16bit	122	207Ah	0
P0123	P1 Contactor 2 Opening Time	0 to 65535 ms	-	ro, 16bit	123	207Bh	0
P0124	P2 Contactor 1 Closing Time	0 to 65535 ms	-	ro, 16bit	124	207Ch	0
P0125	P2 Contactor 1 Opening Time	0 to 65535 ms	-	ro, 16bit	125	207Dh	0
P0126	P2 Contactor 2 Closing Time	0 to 65535 ms	-	ro, 16bit	126	207Eh	0
P0127	P2 Contactor 2 Opening Time	0 to 65535 ms	-	ro, 16bit	127	207Fh	0
P0128	P3 Contactor 1 Closing Time	0 to 65535 ms	-	ro, 16bit	128	2080h	0
P0129	P3 Contactor 1 Opening Time	0 to 65535 ms	-	ro, 16bit	129	2081h	0
P0130	P3 Contactor 2 Closing Time	0 to 65535 ms	-	ro, 16bit	130	2082h	0
P0131	P3 Contactor 2 Opening Time	0 to 65535 ms	-	ro, 16bit	131	2083h	0
P0132	P4 Contactor 1 Closing Time	0 to 65535 ms	-	ro, 16bit	132	2084h	0
P0133	P4 Contactor 1 Opening Time	0 to 65535 ms	-	ro, 16bit	133	2085h	0
P0134	P4 Contactor 2 Closing Time	0 to 65535 ms	-	ro, 16bit	134	2086h	0
P0135	P4 Contactor 2 Opening Time	0 to 65535 ms	-	ro, 16bit	135	2087h	0

Parameter	Description	Range of values	Factory setting	Properties	Communication Address	CANopen Index	Sub-Index
P0150	P1 C1 Operation Counter	0 to 65535	-	ro, 16bit	150	2096h	0
P0151	P1 C2 Operation Counter	0 to 65535	-	ro, 16bit	151	2097h	0
P0152	P2 C1 Operation Counter	0 to 65535	-	ro, 16bit	152	2098h	0
P0153	P2 C2 Operation Counter	0 to 65535	-	ro, 16bit	153	2099h	0
P0154	P3 C1 Operation Counter	0 to 65535	-	ro, 16bit	154	209Ah	0
P0155	P3 C2 Operation Counter	0 to 65535	-	ro, 16bit	155	209Bh	0
P0156	P4 C1 Operation Counter	0 to 65535	-	ro, 16bit	156	209Ch	0
P0157	P4 C2 Operation Counter	0 to 65535	-	ro, 16bit	157	209Dh	0
P0200	P1 Status	0 to 65535	-	ro, 16bit	200	20C8h	0
P0201	P1 Status - Contactor	0 to 1	-	ro, 16bit	201	20C9h	0
P0202	P1 Status - Error	0 to 65535	-	ro, 16bit	202	20CAh	0
P0203	P1 Status - Alarm	0 to 65535	-	ro, 16bit	203	20CBh	0
P0204	P2 Status	0 to 65535	-	ro, 16bit	204	20CCh	0
P0205	P2 Status - Contactor	0 to 1	-	ro, 16bit	205	20CDh	0
P0206	P2 Status - Error	0 to 65535	-	ro, 16bit	206	20CEh	0
P0207	P2 Status - Alarm	0 to 65535	-	ro, 16bit	207	20CFh	0
P0208	P3 Status	0 to 65535	-	ro, 16bit	208	20D0h	0
P0209	P3 Status - Contactor	0 to 1	-	ro, 16bit	209	20D1h	0
P0210	P3 Status - Error	0 to 65535	-	ro, 16bit	210	20D2h	0
P0211	P3 Status - Alarm	0 to 65535	-	ro, 16bit	211	20D3h	0
P0212	P4 Status	0 to 65535	-	ro, 16bit	212	20D4h	0
P0213	P4 Status - Contactor	0 to 1	-	ro, 16bit	213	20D5h	0
P0214	P4 Status - Error	0 to 65535	-	ro, 16bit	214	20D6h	0
P0215	P4 Status - Alarm	0 to 65535	-	ro, 16bit	215	20D7h	0
		S3 Status\Errors and Alarms					
P0300	P1 - Last Error 1	0 to 65535	-	ro, 16bit	300	212Ch	0
P0301	P1 - Last Error 2	0 to 65535	-	ro, 16bit	301	212Dh	0
P0302	P1 - Last Error 3	0 to 65535	-	ro, 16bit	302	212Eh	0
P0303	P1 - Last Error 4	0 to 65535	-	ro, 16bit	303	212Fh	0
P0304	P1 - Last Error 5	0 to 65535	-	ro, 16bit	304	2130h	0
P0305	P2 - Last Error 1	0 to 65535	-	ro, 16bit	305	2131h	0
P0306	P2 - Last Error 2	0 to 65535	-	ro, 16bit	306	2132h	0
P0307	P2 - Last Error 3	0 to 65535	-	ro, 16bit	307	2133h	0
P0308	P2 - Last Error 4	0 to 65535	-	ro, 16bit	308	2134h	0
P0309	P2 - Last Error 5	0 to 65535	-	ro, 16bit	309	2135h	0
P0310	P3 - Last Error 1	0 to 65535	-	ro, 16bit	310	2136h	0
P0311	P3 - Last Error 2	0 to 65535	-	ro, 16bit	311	2137h	0
P0312	P3 - Last Error 3	0 to 65535	-	ro, 16bit	312	2138h	0
P0313	P3 - Last Error 4	0 to 65535	-	ro, 16bit	313	2139h	0
P0314	P3 - Last Error 5	0 to 65535	-	ro, 16bit	314	213Ah	0
P0315	P4 - Last Error 1	0 to 65535	-	ro, 16bit	315	213Bh	0
P0316	P4 - Last Error 2	0 to 65535	-	ro, 16bit	316	213Ch	0
P0317	P4 - Last Error 3	0 to 65535	-	ro, 16bit	317	213Dh	0
P0318	P4 - Last Error 4	0 to 65535	-	ro, 16bit	318	213Eh	0

Parameter	Description	Range of values	Factory setting	Properties	Communication Address	CANopen Index	Sub-Index
P0319	P4 - Last Error 5	0 to 65535	-	ro, 16bit	319	213Fh	0
P0320	P1 - Last Alarm 1	0 to 65535	-	ro, 16bit	320	2140h	0
P0321	P1 - Last Alarm 2	0 to 65535	-	ro, 16bit	321	2141h	0
P0322	P1 - Last Alarm 3	0 to 65535	-	ro, 16bit	322	2142h	0
P0323	P1 - Last Alarm 4	0 to 65535	-	ro, 16bit	323	2143h	0
P0324	P1 - Last Alarm 5	0 to 65535	-	ro, 16bit	324	2144h	0
P0325	P2 - Last Alarm 1	0 to 65535	-	ro, 16bit	325	2145h	0
P0326	P2 - Last Alarm 2	0 to 65535	-	ro, 16bit	326	2146h	0
P0327	P2 - Last Alarm 3	0 to 65535	-	ro, 16bit	327	2147h	0
P0328	P2 - Last Alarm 4	0 to 65535	-	ro, 16bit	328	2148h	0
P0329	P2 - Last Alarm 5	0 to 65535	-	ro, 16bit	329	2149h	0
P0330	P3 - Last Alarm 1	0 to 65535	-	ro, 16bit	330	214Ah	0
P0331	P3 - Last Alarm 2	0 to 65535	-	ro, 16bit	331	214Bh	0
P0332	P3 - Last Alarm 3	0 to 65535	-	ro, 16bit	332	214Ch	0
P0333	P3 - Last Alarm 4	0 to 65535	-	ro, 16bit	333	214Dh	0
P0334	P3 - Last Alarm 5	0 to 65535	-	ro, 16bit	334	214Eh	0
P0335	P4 - Last Alarm 1	0 to 65535	-	ro, 16bit	335	214Fh	0
P0336	P4 - Last Alarm 2	0 to 65535	-	ro, 16bit	336	2150h	0
P0337	P4 - Last Alarm 3	0 to 65535	-	ro, 16bit	337	2151h	0
P0338	P4 - Last Alarm 4	0 to 65535	-	ro, 16bit	338	2152h	0
P0339	P4 - Last Alarm 5	0 to 65535	-	ro, 16bit	339	2153h	0
		S4 Status\Communication					
P0030	CAN - Address	1 to 127	-	ro, 16bit	30	201Eh	0
P0031	CAN - Baudrate	0 = 1 Mbit/s 1 = 800 Kbit/s 2 = 500 Kbit/s 3 = 250 Kbit/s 4 = 125 Kbit/s 5 = 100 Kbit/s 6 = 50 Kbit/s 7 = 20 Kbit/s	-	ro, enum	31	201Fh	0
P0033	CAN - Controller Status	0 = Disabled 1 = Reserved 2 = CAN Enabled 3 = Warning 4 = Error Passive 5 = Bus Off	-	ro, enum	33	2021h	0
P0034	CAN - RX CAN Telegrams	0 to 65535	-	ro, 16bit	34	2022h	0
P0035	CAN - TX CAN Telegrams	0 to 65535	-	ro, 16bit	35	2023h	0
P0036	CAN - Bus Off counter	0 to 65535	-	ro, 16bit	36	2024h	0
P0037	CAN - Lost Telegrams	0 to 65535	-	ro, 16bit	37	2025h	0
P0038	CAN - CANopen Communication Status	0 = Disabled 1 = Reserved	-	ro, enum	38	2026h	0

Parameter	Description	Range of values	Factory setting	Properties	Communication Address	CANopen Index	Sub-Index
		2 = Comm Enabled 3 = Error Ctrl. Enab. 4 = Guarding Error 5 = Heartbeat Error					
P0039	CAN - CANopen Slave Status	0 = Disabled 1 = Initialization 2 = Stopped 3 = Operational 4 = PreOperational	-	ro, enum	39	2027h	0
C1 Configuration\Starts							
P0400	P1 - Operating Mode	0 = Starter 1 = Transparent	0	rw, 16bit	400	2190h	0
P0401	P2 - Operating Mode	0 = Starter 1 = Transparent	0	rw, 16bit	401	2191h	0
P0402	P3 - Operating Mode	0 = Starter 1 = Transparent	0	rw, 16bit	402	2192h	0
P0403	P4 - Operating Mode	0 = Starter 1 = Transparent	0	rw, 16bit	403	2193h	0
P0404	P1 - Contactor Timeout	20 to 5000 ms	500 ms	rw, 16bit	404	2194h	0
P0405	P2 - Contactor Timeout	20 to 5000 ms	500 ms	rw, 16bit	405	2195h	0
P0406	P3 - Contactor Timeout	20 to 5000 ms	500 ms	rw, 16bit	406	2196h	0
P0407	P4 - Contactor Timeout	20 to 5000 ms	500 ms	rw, 16bit	407	2197h	0
P1000	Reset to Factory Settings	0 to 65535	0	rw, 16bit	1000	23E8h	0
C2 Configuration\Communication							
P0101	Action for Communication Fault	0 = No Action 1 = Disable Outputs	1	rw, enum	101	2065h	0
P0032	CAN - Bus Off Reset	0 = Manual 1 = Automatic	0	rw, enum	32	2020h	0
C3 Configuration\Counters							
P0158	Save Operation Counters in the mem NV	FALSE to TRUE	FALSE	rw, bool	158	209Eh	0
P0160	Reset P1 C1 Operation Counter	0 to 65535	0	rw, 16bit	160	20A0h	0
P0161	Reset P1 C2 Operation Counter	0 to 65535	0	rw, 16bit	161	20A1h	0
P0162	Reset P2 C1 Operation Counter	0 to 65535	0	rw, 16bit	162	20A2h	0
P0163	Reset P2 C2 Operation Counter	0 to 65535	0	rw, 16bit	163	20A3h	0
P0164	Reset P3 C1 Operation Counter	0 to 65535	0	rw, 16bit	164	20A4h	0
P0165	Reset P3 C2 Operation Counter	0 to 65535	0	rw, 16bit	165	20A5h	0
P0166	Reset P4 C1 Operation Counter	0 to 65535	0	rw, 16bit	166	20A6h	0
P0167	Reset P4 C2 Operation Counter	0 to 65535	0	rw, 16bit	167	20A7h	0
C4 Configuration\Commands							

Parameter	Description	Range of values	Factory setting	Properties	Communication Address	CANopen Index	Sub-Index
P1100	Forward Start Command	Bit 0 = Start 1 - run forward Bit 1 = Start 2 - run forward Bit 2 = Start 3 - run forward Bit 3 = Start 4 - run forward	0	rw, 16bit	1100	244Ch	0
P1105	Reverse Start Command	Bit 0 = Start 1 - run forward Bit 1 = Start 2 - run forward Bit 2 = Start 3 - run forward Bit 3 = Start 4 - run forward	0	rw, 16bit	1105	2451h	0
P1110	Stop Command	Bit 0 = Start 1 - stop Bit 1 = Start 2 - stop Bit 2 = Start 3 - stop Bit 3 = Start 4 - stop	0	rw, 16bit	1110	2456h	0
P1115	Digital Output Command	Bit 0 = DO1 Bit 1 = DO2 Bit 2 = DO3 Bit 3 = DO4 Bit 4 = DO5 Bit 5 = DO6 Bit 6 = DO7 Bit 7 = DO8	0	rw, 16bit	1115	245Bh	0

Table A.3: *Description of the parameter data types*

Data Type	Description
bool	Represents a bit where the value 0 (zero) represents false, the value 1 (one) represents true.
enum	Enumerated type (unsigned 8-bit) contains a list of values with function description for each item.
8bit	Unsigned 8-bit integer, ranges from 0 to 255.
16bit	Unsigned 16-bit integer, ranges from 0 to 65,535.
s16bit	Signed 16-bit integer, ranges from -32,768 to 32,767.
32bit	Unsigned 32-bit integer, ranges from 0 to 4,294,967,295.

WEG Drives & Controls - Automação LTDA.
Jaraguá do Sul – SC – Brazil
Phone 55 (47) 3276-4000 – Fax 55 (47) 3276-4020
São Paulo – SP – Brazil
Phone 55 (11) 5053-2300 – Fax 55 (11) 5052-4212
automacao@weg.net
www.weg.net