CANopen

CFW100

User's Manual

CANopen User's Manual

Series: CFW100

Language: English

Document Number: 10002835377 / 00

Publication Date: 06/2014

CONTENTS

ABOUT THE MANUAL

This manual provides the necessary information for the operation of the CFW100 frequency inverter using the CANopen protocol. This manual must be used together with the CFW100 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 CFW100 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 CFW100. 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 frequency inverter CFW100 is programmed by .

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 CFW100 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 CFW100 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

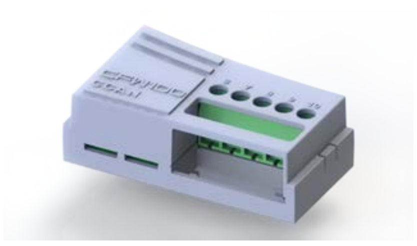
Each device in a CANopen network has an EDS configuration file that contains information about the operation of the device in the CANopen network, as well as the description of all the communication objects available. In general, this file is used by a master or by the configuration software for programming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

The EDS configuration file for the CFW100 is supplied together with the product, and it can also be obtained from the website. 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 ACCESSORY

In order to make the CANopen communication possible with the product, it is necessary to use the CAN communication kit described next. Information on the installation of this module can be obtained in the guide that comes with the kit.

2.1 PLUG-IN MODULE CFW500-CCAN



- WEG part number: 12293349.
- Composed by the CAN communication module (drawing at the left), mounting instruction and fixing screw.
- The interface is electrically isolated and with differential signal, which grants more robustness against electromagnetic interference.
- External 24V supply.
- It allows the connection of up to 64 devices to the same segment. More devices can be connected by using repeaters[3].
- A maximum bus length of 1000 meters.

2.2 CONNECTOR PINOUT

The CAN communication module presents a 5-wire connector with the following pinout:

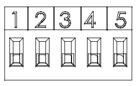


Table 2.1: *CAN interface connector pinout*

Pin	Name	Function
1	V-	Power supply negative pole
2	CAN_L	CAN_L communication signal
3	Shield	Cable shield
4	CAN_H	CAN_H communication signal
5	V+	Power supply positive pole

NOTE!
It is recommended to connect the GND pin of the CFW500-CCAN module to protective earth. This is necessary to connect the cable shield with the earth.

2.3 POWER SUPPLY

The CAN interface needs an external power supply between the pins 1 and 5 of the network connector. The individual consumption and input voltage data are presented in the next table.

Table 2.2: *CAN interface supply characteristics*

Supply Voltage (V_{DC})		
Minimum	Maximum	Recommended
11	30	24
Current (mA)		
Typical	Maximum	
30	50	

2.4 INDICATIONS

Details on the alarms, communications failures and communication states are made through the keypad (HMI) and product parameters.

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

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 CFW100 frequency inverter the baud rate configuration is done through the .

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 CFW100 frequency inverter the address configuration is done through the .

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.

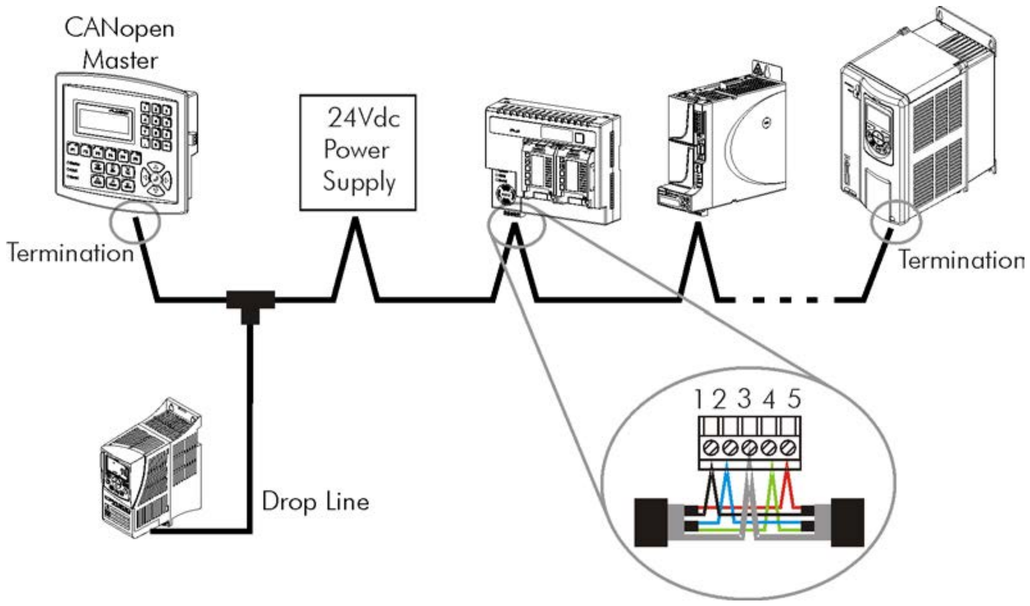


Figure 3.1: *CANopen network installation example*

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 CFW100 frequency inverter 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

P0105 – 1ST/2ND RAMP SELECTION

P0220 – LOCAL/REMOTE SELECTION SOURCE

P0221 – SPEED REFERENCE SELECTION – LOCAL SITUATION

P0222 – SPEED REFERENCE SELECTION – REMOTE SITUATION

P0223 – FORWARD/REVERSE SELECTION – LOCAL SITUATION

P0224 – RUN/STOP SELECTION – LOCAL SITUATION

P0225 – JOG SELECTION – LOCAL SITUATION

P0226 – FORWARD/REVERSE SELECTION – REMOTE SITUATION

P0227 – RUN/STOP SELECTION – REMOTE SITUATION

P0228 – JOG SELECTION – REMOTE SITUATION

These parameters are used in the configuration of the command source for the CFW100 frequency inverter local and remote situations. In order that the device be controlled through the CANopen interface, the options 'CANopen/DeviceNet/Profibus DP' available in these parameters, must be selected.

The detailed description of these parameters is found in the CFW100 programming manual.

P0313 – COMMUNICATION ERROR ACTION

Range:	0 = Inactive 1 = Disable via Run/Stop 2 = Disable via General Enable 3 = Change to Local 4 = Change to Local keeping commands and reference 5 = Causes a Fault	**Default:** 1
Properties:	CFG	

Description:
It allows the selection of the action to be executed by the device, if it is controlled via network and a communication error is detected.

Table 4.1: *P0313 options*

Options	Description
0 = Inactive	No action is taken and the drive remains in the existing status.
1 = Disable via Run/Stop	A stop command with deceleration ramp is executed and the motor stops according to the programmed deceleration ramp.
2 = Disable via General Enable	The drive is disabled by removing the General Enabling and the motor coasts to stop.
3 = Change to Local	The drive commands change to Local.
5 = Causes a Fault	Instead of an alarm, the communication error causes a drive fault, so that a drive fault reset becomes necessary in order to restore normal operation.

The following events are considered communication errors:

CANopen communication:
- A133 alarm/F233 fault: CAN interface not powered.
- A134 alarm/F234 fault: *bus off*.
- A135 alarm/F235 fault: CANopen communication error (*Node Guarding/Heartbeat*).

The actions described in this parameter are executed by means of the automatic writing of the selected actions in the respective bits of the interface control words. Therefore, in order that the commands written in this parameter be effective, it is necessary that the device be programmed to be controlled via the used network interface (with exception of option "Causes a Fault", which blocks the equipment even if it is not controlled by network). This programming is achieved by means of parameters P0220 to P0228.

P0680 – STATUS WORD

Range:	0000h to FFFFh	**Default:** -
Properties:	RO	

Description:
It allows the device status monitoring. Each bit represents a specific status:

Bits	15	14	13	12	11	10	9	8	7	6	5	4 to 0
Function	Fault condition	Reserved	Undervoltage	LOC/REM	JOG	Speed direction	Active General Enable	Motor Running	Alarm condition	In configuration mode	Second ramp	Reserved

Table 4.2: *P0680 parameter bit functions*

Bits	Values
Bits 0 to 4	Reserved.
Bit 5 Second ramp	0: The drive is configured to use the first ramp values, programmed in P0100 and P0101, as the motor acceleration and deceleration ramp times. 1: The drive is configured to use the second ramp values, programmed in P0102 and P0103, as the motor acceleration and deceleration ramp times.
Bit 6 In configuration mode	0: The drive is operating normally. 1: The drive is in the configuration mode. It indicates a special condition during which the drive cannot be enabled: Executing the self-tuning routine Executing the oriented start-up routine Executing the HMI copy function Executing the flash memory card self-guided routine There is a parameter setting incompatibility There is no power at the drive power section
Bit 7 Alarm condition	0: The drive is not in alarm condition. 1: The drive is in alarm condition. Note: The alarm number can be read by means of the parameter P0048 – Present Alarm.
Bit 8 Motor Running	0: The motor is stopped. 1: The drive is running the motor at the set point speed, or executing either the acceleration or the deceleration ramp.
Bit 9 Active General Enable	0: General Enable is not active. 1: General Enable is active and the drive is ready to run the motor.
Bit 10 Speed direction	0: The motor is running in the reverse direction. 1: The motor is running in the forward direction.
Bit 11 JOG	0: Inactive JOG function. 1: Active JOG function.
Bit 12 LOC/REM	0: Drive in Local mode. 1: Drive in Remote mode.
Bit 13 Undervoltage	0: No Undervoltage. 1: With Undervoltage.
Bit 14	Reserved.
Bit 15 Fault condition	0: The drive is not in a fault condition. 1: The drive has detected a fault. Note: The fault number can be read by means of the parameter P0049 – Present Fault.

P0681 – MOTOR SPEED IN 13 BITS

Range: - 32768 to 32767 **Default:** -
Properties: RO

Description:
It allows monitoring the motor speed. This word uses 13-bit resolution with signal to represent the motor rated frequency (P0403):

- P0681 = 0000h (0 decimal) $\rightarrow$ motor speed = 0
- P0681 = 2000h (8192 decimal) $\rightarrow$ motor speed = rated frequency

Intermediate or higher speed values in rpm can be obtained by using this scale. E.g.60Hz rated frequency if the value read is 2048 (0800h), then, to obtain the speed in Hz one must calculate:

8192 => 60 Hz
2048 => Frequency in Hz

$$\text{Frequency in Hz} = \frac{60 \times 2048}{8192}$$

Frequency in Hz = 15 Hz

Negative values in this parameter indicate that the motor is running in the reverse direction.

P0684 – CANOPEN CONTROL WORD

Range:	0000h to FFFFh	**Default:** 0000h
Properties:	-	

Description:
It is the device CANopen interface control word. This parameter can only be changed via CANopen/DeviceNet/Profibus DP interface. For the other sources (HMI, etc.) it behaves like a read-only parameter.

In order to have those commands executed, it is necessary to program the equipment to be controlled via CANopen/DeviceNet/Profibus DP. This programming is achieved by means of parameters P0105 and P0220 to P0228.

Each bit of this word represents a command that can be executed.

Bits	**15 to 8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Function	Reserved	Fault reset	Reserved	Second ramp	LOC/REM	JOG	Speed direction	General enable	Run/Stop

Table 4.3: *P0684 parameter bit functions*

Bits	**Values**
Bit 0 Run/Stop	0: It stops the motor with deceleration ramp. 1: The motor runs according to the acceleration ramp until reaching the speed reference value.
Bit 1 General enable	0: It disables the drive, interrupting the supply for the motor. 1: It enables the drive allowing the motor operation.
Bit 2 Speed direction	0: To run the motor in a direction opposed to the speed reference. 1: To run the motor in the direction indicated by the speed reference.
Bit 3 JOG	0: It disables the JOG function. 1: It enables the JOG function.
Bit 4 LOC/REM	0: The drive goes to the Local mode. 1: The drive goes to the Remote mode.
Bit 5 Second ramp	0: The drive uses the first ramp values, programmed in P0100 and P0101, as the motor acceleration and deceleration ramp times. 1: The drive is configured to use the second ramp values, programmed in P0102 and P0103, as the motor acceleration and deceleration ramp times.
Bit 6	Reserved.
Bit 7 Fault reset	0: No function. 1: If in a fault condition, then it executes the reset.
Bits 8 to 15	Reserved.

P0685 – CANOPEN SPEED REFERENCE

Range:	-32768 to 32767	**Default:** 0
Properties:	-	

Description:
It allows programming the motor speed reference via the CANopen interface. This parameter can only be changed via CANopen/DeviceNet/Profibus DP interface. For the other sources (HMI, etc.) it behaves like a read-only parameter.

In order that the reference written in this parameter be used, it is necessary that the drive be programmed to use the speed reference via CANopen/DeviceNet/Profibus DP. This programming is achieved by means of parameters P0221 and P0222.

This word uses a 13-bit resolution with signal to represent the motor rated frequency (P0403).

- P0685 = 0000h (0 decimal) → speed reference = 0
- P0685 = 2000h (8192 decimal) → speed reference = rated frequency (P0403)

Intermediate or higher reference values can be programmed by using this scale. E.g.60Hz rated frequency, to obtain a speed reference of 30 Hz one must calculate:

```
60 Hz => 8192
30 Hz => 13 bit reference
```

$$\text{13 bit reference} = \frac{30 \times 8192}{60}$$

13 bit reference = 4096 => Value corresponding to 30 Hz in a 13 bit scale

This parameter also accepts negative values to revert the motor speed direction. The reference speed direction, however, depends also on the control word - P0684 - bit 2 setting:

- Bit 2 = 1 and P0685 > 0: reference for forward direction
- Bit 2 = 1 and P0685 < 0: reference for reverse direction
- Bit 2 = 0 and P0685 > 0: reference for reverse direction
- Bit 2 = 0 and P0685 < 0: reference for forward direction

P0700 – CAN PROTOCOL

Range:	1 = CANopen 2 = DeviceNet	**Default:** 2
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 1 = Reserved 2 = 500 Kbit/s 3 = 250 Kbit/s 4 = 125 Kbit/s 5 = 100 Kbit/s 6 = 50 Kbit/s 7 = Reserved 8 = Reserved	**Default:** 0
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:** 0
Properties:		

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

Table 4.4: *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 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 .

P0705 – CAN CONTROLLER STATUS

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

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

Table 4.5: *Values for the parameter P0705*

Value	Description
0 = Disabled	Inactive CAN interface. It occurs when the equipment does not have the CAN interface installed.
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.

P0706 – 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.

P0707 – 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.

P0708 – 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.

P0709 – 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.

P0721 – 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*).

P0722 – 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.

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 frequency inverter CFW100.is presented next.

Table 5.2: *Object list – Communication Profile*

Index	Object	Name	Type	Access
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
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
Transmit PDO Mapping Parameter				
1A00h	RECORD	1st transmit PDO mapping	PDO Mapping	rw
1A01h	RECORD	2nd 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 – CFW100 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 CFW100, the whole list of parameters was made available in this object range. It is possible to operate the CFW100 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: *CFW100 object list – Manufacturer Specific*

Index	Object	Name	Type	Access
2000h	VAR	P0000 – Access parameter	INTEGER16	rw
2001h	VAR	P0001 – Speed reference	INTEGER16	ro
2002h	VAR	P0002 – Motor speed	INTEGER16	ro
2003h	VAR	P0003 – Motor current	INTEGER16	ro
2004h	VAR	P0004 – DC voltage	INTEGER16	ro
...	...	...	...	...
2064h	VAR	P0100 – Acceleration time	INTEGER16	rw
2065h	VAR	P0101 – Deceleration time	INTEGER16	rw
...	...	...	...	...

Refer to the CFW100 manual for a complete list of the parameters and their detailed description. In order to be able to program the inverter operation correctly via the CANopen network, it is necessary to know its operation through the parameters.

6 COMMUNICATION OBJECTS DESCRIPTION

This item describes in detail each of the communication objects available for the frequency inverter CFW100 . It is necessary to know how to operate these objects to be able to use the available functions for the inverter communication.

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	

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 CFW100 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	

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 CFW100

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 frequency inverter CFW100 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.

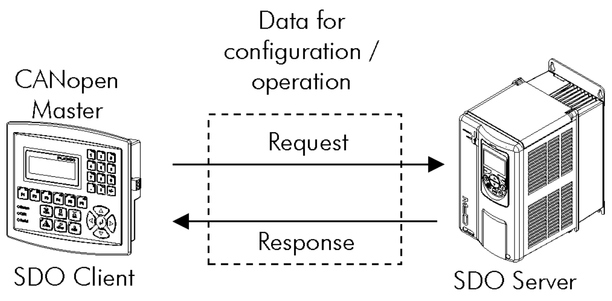


Figure 6.1: *Communication between SDO client and server*

6.2.1 Object 1200h – SDO Server

The frequency inverter CFW100 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.

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 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.

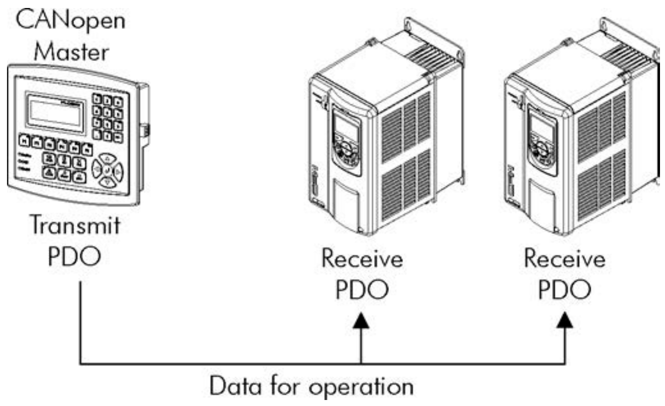


Figure 6.2: *Communication using PDOs*

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.

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 frequency inverter CFW100 has 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 140h
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

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 6.4: *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 CFW100 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 6.5: *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 160h
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 ... = number of mapped objects
Default value	0

Sub index	1 up to
Description	1 up to 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 CFW100 receive PDOs. It is possible to map up to 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[5] 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:

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

[5] 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.

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 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 CFW100 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 frequency inverter CFW100 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

Index	1800h up to 180h
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

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 6.4.

The sub-index 2 indicates the transmission type of this object, which follows the table 6.5 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 frequency inverter CFW100 is .
- 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 1A0h
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 ... = number of mapped objects
Default value	0

Sub index	1 up to
Description	1 up to 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:

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 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 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 frequency inverter CFW100 has the function of a SYNC consumer and, therefore, it can program its PDOs to be synchronous. As described in table 6.5, synchronous PDOs are those related to the synchronization object, thus they can be programmed to be transmitted or updated based in this object.

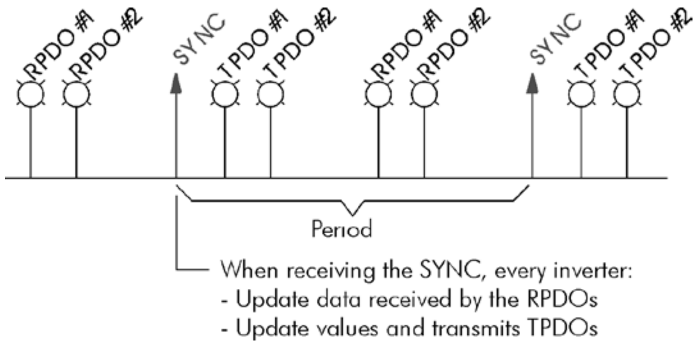

RPDO #1
RPDO #2
SYNC
TPDO #1
TPDO #2
RPDO #1
RPDO #2
SYNC
TPDO #1
TPDO #2
Period
When receiving the SYNC, every inverter:
- Update data received by the RPDOs
- Update values and transmits TPDOs

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 CFW100 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 CFW100 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:

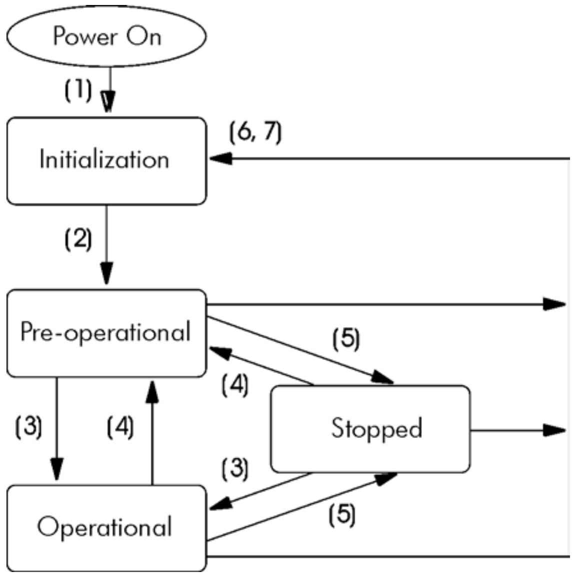


Figure 6.4: *CANopen node state diagram*

Table 6.6: *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 6.7: *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 6.8: *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 CFW100 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.

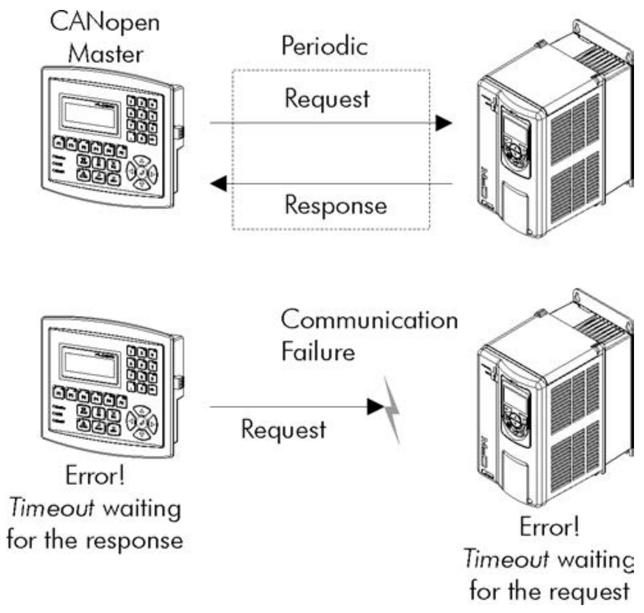

CANopen Master
Periodic
Request
Response
Communication Failure
Request
Error! Timeout waiting for the response
Error! Timeout waiting for the request

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 CFW100 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 CFW100 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*).

NOTE!
- This object is active even in the stopped state (see table 6.7).
- 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 CFW100 is ., 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.

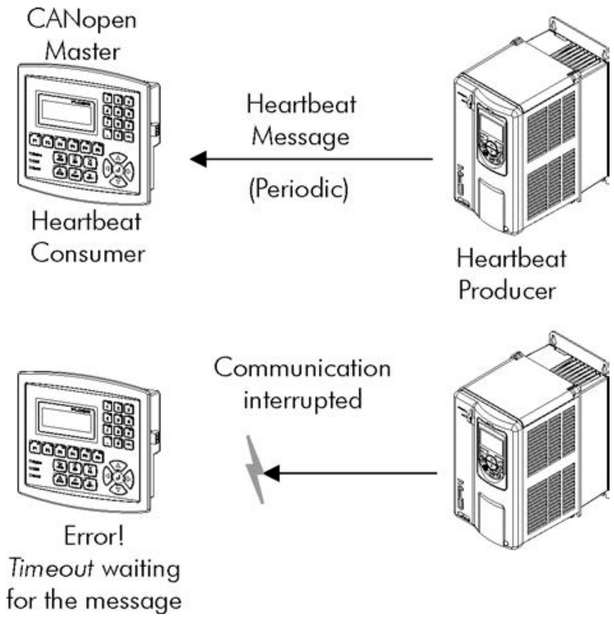


Figure 6.6: *Error control service – Heartbeat*

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 CFW100 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	

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

At sub-indexes 1 to , 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 frequency inverter will turn automatically to the preoperational state and indicate .

As a producer, the frequency inverter CFW100 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 6.7).
- 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 frequency inverter CFW100 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:

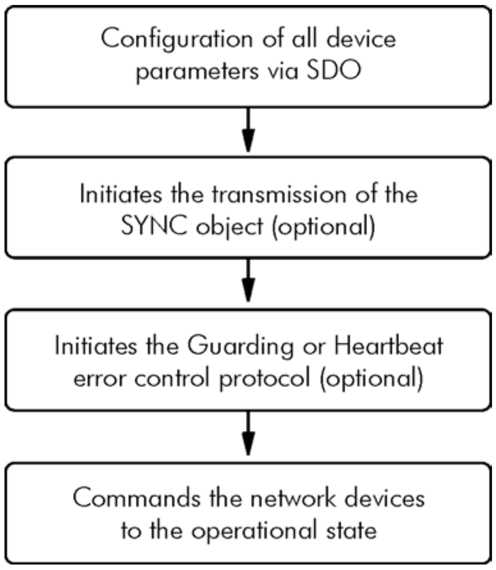

Configuration of all device parameters via SDO
Initiates the transmission of the SYNC object (optional)
Initiates the Guarding or Heartbeat error control protocol (optional)
Commands the network devices to the operational state

Figure 6.7: *Initialization process flowchart*

It is necessary to observe that the frequency inverter CFW100 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.

7 FAULTS AND ALARMS RELATED TO THE CANOPEN COMMUNICATION

A133/F233 – 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 F233, depending on the P0313 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/F234 – 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[6], 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 F234, depending on the P0313 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/F235 – 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.

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

In this case the alarm A135 or the fault F235, depending on the P0313 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.