PLC2 Board
Communication Manual
CANopen Slave

02/2006

Series: PLC2
0899.5809 E/3

Contents

List of Tables

List of Figures

About the Manual

This manual describes the operation of the CANopen protocol for the PLC2 board. This manual must be used along with the PLC2 Board 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
EMCY	Emergency Object
SYNC	Synchronization Object
NMT	Network Management Object
ASCII	American Standard Code for Information Interchange

Numerical representation
The decimal numbers are represented by digits without suffix. The hexadecimal numbers are represented by the letter 'h' after the number.

Documents

The PLC2 CANopen protocol development has been 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

Table 1: Technical documents about CANopen

Please contact CiA (CAN in Automation) for providing these documents. This company owns, releases and updates all information relating to the CANopen network.

1 Introduction to the CANopen protocol

For proper PLC2 board operation on network you must know how this communication is realized. Thus this section gives a general description of the CANopen protocol operation by describing all functions used by the PLC2. For more details about the protocol, refer to CANopen documents indicated in table 1.

1.1 CAN

The CANopen network is based on the CAN network, which means that it uses CAN telegrams for the network data exchange.

The CAN protocol is a serial communication protocol that describes the services from the layer 2 of the ISO / OSI model (layer of the data link)[1]. In this layer are defined the different telegram types (frames), how this errors are detected and how the message validation and arbitration are realized.

1.1.1 Data frame

The CAN network data are transmitted through a data frame (telegram). This frame type comprises several fields, like start of frame, control field, CRC, etc. However the main fields are an 11 bits identifier field[2] (arbitration field) and a data field which 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

In addition the data frame there is a remote frame (RTR frame). This frame type does not have data field, but only the identifier. It requests the other network device for transmitting the desired data frame.

1.1.3 Network access

Any network element on a CAN network can try to transmit at any time a frame to the network. If two network devices try to access the network simultaneously, only the device transmitting the top priority message will have success in transmitting the message to the network. The message priority is defined by the CAN frame identifier. The lesser the value of this identifier is, the higher the priority of the message. The telegram with identifier 0 (zero) corresponds to the telegram with the top priority.

[1]In the CAN protocol specification is referenced the standard ISO 11898 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 bits) and extended (28 bits). For the CANopen protocol of the PLC2 board only standard frames are accepted.

1.1.4 Error control

The CAN specification defines several error control mechanisms, ensuring high network reliability and low transmission error index that are not detected. Each network device must be able to identify this error occurrence, informing the other elements that an error has been detected.

A CAN network device has internal counters which are incremented every time a transmission or reception error has been detected and decremented when a telegram has been received with success. When a significant number of errors are detected, the device may enter into the following states:

- *Warning*: the device enters into the warning state when the counter exceeds a determined error limit, which means that a high error index has been detected.
- *Error Passive*: when this value exceeds the established limit, the device enters into the error passive, and then stops acting on the network when it detects that another device has sent a telegram with error.
- *Bus Off*: at least, we will have the bus off state, and the device will not send nor receive telegrams more.

1.1.5 CAN and CANopen

The specification on how detecting errors, generating and transmitting a frame is not sufficient for defining a meaning for the data that should be sent via network. A specification is required that indicates how the identifier and the data should be mounted and how the information should be exchanged and so enabling the network elements for interpreting the transferred data correctly. So the CANopen specification defines how the data exchange between equipments is realized and how each device should interpret these data.

There are also other CAN based protocols, as the DeviceNet, J1939, etc. that use CAN frames for the communication. However these protocols can not operate jointly on the same network.

1.2 Features of the CANopen network

As the telegram transfer is realized through a CAN bus, all CANopen network devices have the same right for accessing the network, where the priority of the identifier is responsible for solving the conflict problems when accesses occur simultaneously. This will be useful, since it enables the communication between network slaves, in addition to the fact that the data are available more optimized without requiring a master for controlling all communication, realizing cyclic access to all network devices for data updating.

Other important feature of the CANopen network is the use of a producer / consumer model for the data transmission. This means that the message that is transmitted via network does not have a fixed destination address. The message has an identifier that identifies which data that is being transported. Any network element that requires using this information for its logic operation can consume it. Thus the same message can be used at the same time by several elements.

1.3 Physical layer

The physical layer for signal transmitting on CANopen network is specified in the standard ISO 11898. This standard defines a twisted pair of wire with differential electrical signal for the transmission bus.

The PLC2 board also uses an interface circuit isolated against power line with external power supply. The component responsible for the signal transmission and reception is designated transceiver and complies with the ISO 11898 specification.

1.4 Address at CANopen network

Every CANopen network must be fitted with a master, that is responsible for the network managing and it can have up to 127 slaves. The network device may also be called node. Every slave at a CANopen network is identified on the network by its address, or Node-ID, which must be exclusive for each slave on the network and can be from 1 up to 127.

The PLC2 board does not have functions that implement the network managing services. Thus it must be used along with any equipment that offers such service.

1.5 Data access

Every slave on the CANopen network has a list, designated as object dictionary, where are indicated all data that can be accessed via network. Every object in this listing is identified by an index and this index is used during the equipment configuration and message exchange for identifying which object is being transmitted.

For more details on how this dictionary is structured, refer to section 4.

1.6 Data transfer

The transmission of numerical data through CANopen telegrams is realized by using the hexadecimal number presentation and the least significant data byte is sent firstly.

Example: transmission through a CAN frame of a 32 bits signed integer (12345678h = 305419896 decimal) and a 16 bits signed integer (FF00h = -256 decimal).

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

1.7 Communication objects - COBs

There is a fixed set of objects which are responsible for the communication between the network devices. The objects are divided according to the data types and the way they are sent or received by the device. The PLC2 board supports following communication objects (COBs):

Object type	***Description***
Service Data Object (SDO)	The SDOs are the objects responsible for the direct access to the object dictionary of a device. Through the messages that use the SDOs, you can indicate explicitly (through the object index), which data is being processed. There are two types of SDOs: SDO Client responsible for read/write request from a network device, and the SDO Server responsible for servicing this request. As the SDOs are used in general for a network node configuration, they have less priority than the other message types. Only one SDO of server type is available for the PLC2.
Process Data Object (PDO)	The PDOs are used for accessing the equipment data without indicating explicitly which dictionary object is being accessed. Thus previous configuration is required for indicating which data the PDO is transmitting (data mapping). There are also two types of PDOs: receive PDO and transmit PDO. As the PDOs are used in general for data transmission and data reception during the device operation, they have more priority than Service Data Objects (SDO).
Emergency Object (EMCY)	This object is responsible for the message sending to indicate the error occurrence on the device. When an error is detected on some device (EMCY Producer), it can send a message to the network. If some network device is monitoring this message (EMCY Consumer), you can program it for enabling an action (enable the other network devices, error reset, etc.). The PLC2 board has the EMCY producer function.
Synchronisation Object (SYNC)	You can program a device (SYNC Producer) on the CANopen network for sending periodically a synchronization message to all network messages. So these devices (SYNC Consumer) can send, for example, determined data which must be available from time to time. The PLC2 board has the SYNC consumer function.
Network Management (NMT)	Every CANopen network requires a master for controlling the other network devices (slaves). This master will be responsible for controlling the slave communication and its status on the CANopen network. The slaves receive the commands sent by the master and execute the requested actions. The PLC2 operates as slave on the CANopen Network and provides two services that can be used by the master: device controlling services, where the master controls the status of each slave on the network, and the error controlling services (node guarding), where both the master and the slave exchange telegrams periodically for checking if there are no communication errors.

Table 2: Communication objects - COBs

The board communication to the network is realized through these objects and the accessible data are those existing in the device object dictionary. For more details about the operation of each COB, refer to section 5. The PLC2 operation model, using these communication objects, is described by the following figure:

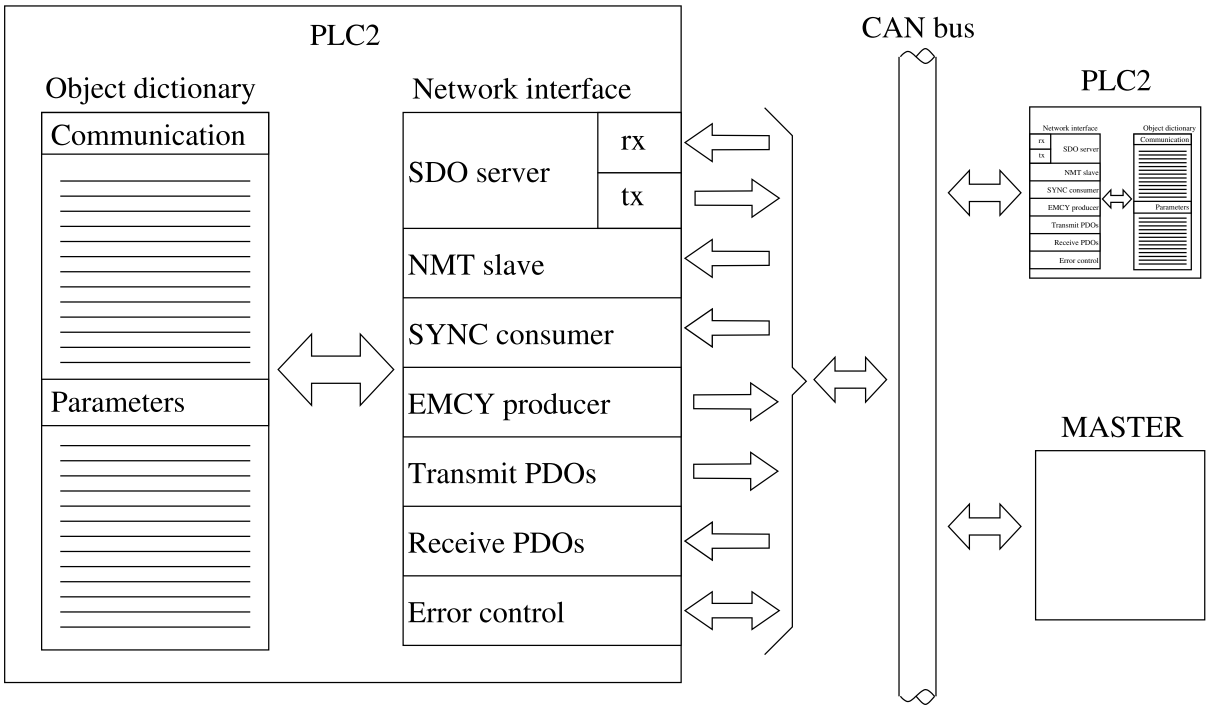


Figure 1: PLC2 operation model on the CANopen network

1.8 COB-ID

The telegram of the CANopen network is always transmitted by a communication object (COB). Each COB has an identifier indicating the data type which is being transmitted. This identifier, called COD-ID, has a length of 11 bits and is transmitted in the identifier field of the CAN telegram. It can be divided into two parts:

Function Code				*Node 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 object type which is being transmitted.
- *Node Address*: indicates to which network device the telegram is linked.

Table below shows the standard values for the different communication objects available on the PLC2 board. Please note that the standard object value depends on the slave address for executing the COB-IDs to NMT and SYNC that are common to all network elements. Some values can be changed during the device configuration process.

COB	***Function Code*** (bits 10 - 7)	***Resulting COB-IDs*** (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	1110	1793 - 1919 (701h - 77Fh)

Table 3: COB-ID for the different objects

1.9 EDS File

Each device in a CANopen network has an EDS configuration file, that contains information about the device function on CANopen network, as well the description of all available objects for communication. This file is most commonly used by the CANopen master or configuration software to program the CANopen slaves in the network.

The EDS configuration file is supplied with the product. It is also possible to download it from http://www.weg.com.br. It is important to pay attention to the PLC2 software version, in order to use the EDS file compatible with this version.

2 Installation

Like most industrial communication networks, the CANopen network requires some special cares for ensuring safe operation with low communication error incidence, when installed in aggressive environments with high exposure to electromagnetic interferences. The installation should be carried out according to ISO 11898. Please find below some hints for correct installation of the PLC2 board.

2.1 Connection to the network

The interface for the CAN bus connection is available at the connector X17. Please find below an installation example, where a PLC2 board is connected to a CFW-09 frequency inverter.

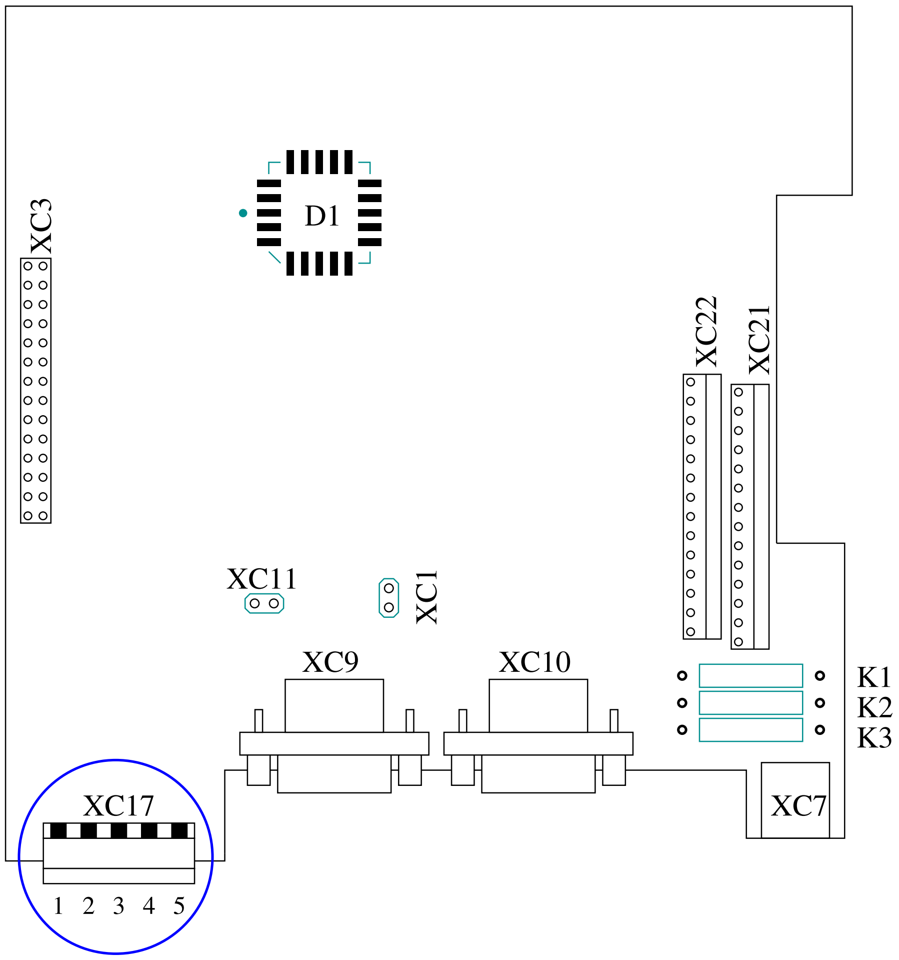
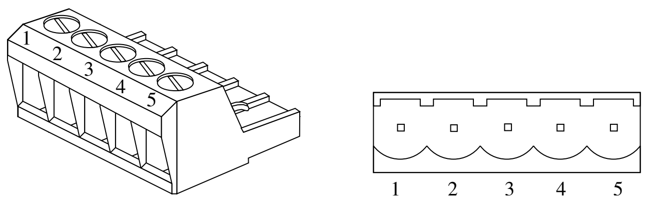


Figure 2: CAN connector location on the PLC2 board

Table 4 describes the function of each connector pin.



Female connector (bus) Male connector (PLC2)

Figure 3: Connectors

X17 connector	
1	V-
2	CAN_L
3	Shield
4	CAN_H
5	V+

Table 4: Connector pinout used by the PLC2

The connection must be carried out by connecting every signal (V - connected to V-, CAN_L connected to CAN_L, etc.) of the different equipments connected on network. If some equipment does not require to be powered via network, do not connect the points V- and V+. The cable shield must be connected to pin 3, which is internally connected to the ground through a RC circuit.

2.2 Power supply

For the power supply of circuit, responsible for the communication on the PLC2 board, you must provide a supply voltage between the pins 1 and 5 of the network connector.

To prevent supply voltage differences between the network devices, we recommend supplying the network at only one point, and this supply voltage should be supplied through the same cable to all devices. If more than one power supply is required, these power supplies should be referenced to the same connection point. Table below shows the individual consumption and the required input voltage.

Power Supply (VDC)		
Minimum	Maximum	Recommended
11	30	24
Current (mA)		
Minimum	Maximum	Average
20	50	30

Table 5: Module consumption

NOTE!
This power supply is used by the electrically isolated CAN components only. In order to communicate with the CANopen network, the drive main power must be on.

2.3 Cables and termination resistor

The use of a shielded cable with two twisted pair wires is recommended - one pair for the pins 2 and 4 (CAN_L and CAN_H) and the other pair for the pins 1 and 5 (V- and V+).

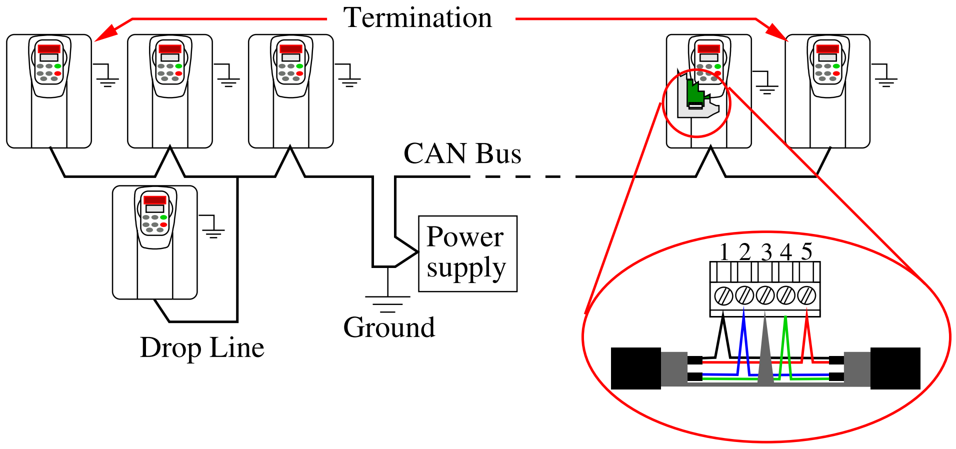


Figure 4: CANopen network

For linkage of several network nodes, we recommend connecting the equipment direct to the main line without drop lines. To avoid electromagnetic interference and ensure trouble free operation, provide physical separation between the power cables and the signal and control cables. To avoid different potentials between the different circuits, ground all equipments to the same grounding point.

The signal cable CAN_L and CAN_H must have a characteristic impedance around 120Ω and a max. signal propagation delay of 5 ns/m. Other cable characteristics depend on its length and must meet data indicated in table below.

Cable length (m)	***Resistance 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

Table 6: Cable characteristics

The CAN bus ends must be fitted with a 120Ω / 0.25W termination resistor connected to the signals CAN_H and CAN_L. When the PLC2 board is the first or last device connected to the bus, this resistor can be fitted directly to the CAN connector, between the pins 2 and 4.

The maximum number of devices connected to an only network segment is limited to 64 devices. If a more devices are required, additional repeaters must be installed.

2.4 Baud rate

The transfer rate that can be used by equipment installed on the CANopen network depends on the cable length used in the installation. Table below shows the transfer rates available for the PLC2 board and the max. allowed cable length as recommended by CiA.

Baud rate	***Cable length***
1 Mbit/s	40 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

Table 7: Supported transfer rate and installation size

3 CANopen communication parameters

The PLC2 board has a set of parameters that allows the network device configuration, the trouble shooting and CANopen communication monitoring.

All not mentioned parameters have not direct relationship with the communication, however they are important for the PLC2 board operation. Thus you must be acquainted with this parameter operation since they may be required for you CANopen network operation.

3.1 P770 - CAN protocol

P770 allows the selection of the application layer protocol required for the CAN bus on the PLC2 board. Select option '1' for enabling the CANopen communication.

Range	*Default*	*Access*
0 = Disabled 1 = CANopen 2 = DeviceNet	0 = Disabled	Read/ write

NOTE!
- This parameter changing will be valid only when the board is reset or switched off and on.
- Different protocols can not operate on the same network.

3.2 P771 - Address of the CAN network

This parameter allows the PLC2 address selection (Node-ID) on the CANopen network.

Range	*Default*	*Access*
1 ... 127	63	Read/ write

Every network device needs a distinct Node-ID, thus up to 127 devices are permitted on an only network (by using repeaters). This Node-ID is also used for defining the initial value of some communication objects on the PLC2 board.

NOTE!
This Node-ID changing will be valid only when the board is reset or switched off and on.

3.3 P772 - Baud rate

This parameter allows the selection of the communication baud rate used by the device.

Range	*Default*	*Access*
0 = 1 Mbit/s 1 = Not used 2 = 500 kbit/s 3 = 250 kbit/s 4 = 125 kbit/s 5 = 100 kbit/s 6 = 50 kbit/s 7 = 20 kbit/s 8 = 10 kbit/s	0 = 1 Mbit/s	Read/ write

You must configure all devices to the same baud rate for enabling the network device communication. Please note that there is a baud rate limitation, depending on the cable length of the installation (see table 7).

NOTE!
The baud rate changing will be valid only when the board is reset or switched off and on.

3.4 P773 - Bus Off recovery

The device may enter into the bus off status (see section 1.1.4) when the number of errors on the CAN network is too high, thus interrupting the network accessing. If this error occurs, the parameter P773 allows programming if the PLC2 remains in the bus off state, requiring manual reset, or it resets automatically and restarts the communication.

Range	*Default*	*Access*
0 = Manual 1 = Automatic	0	Read/ write

3.5 P774 - Action for communication error

If the device, which is controlled by the board, is enabled and a communication error is detected (broken cable, line voltage drop, etc.), you can not send disabling commands to it via network. To avoid this condition, you can program an action at P774, which the PLC2 will execute automatically in case of a network fault.

Range	*Default*	*Access*
0 = No action 1 = Drive disable	1	Read/ write

The PLC2 board detects as CANopen communication errors the bus power failure (E63), bus off (E61) and the timeout conditions during the node guarding service (E65). In case of bus off event, the board only consider it as an error if the bus off recovery is programmed to manual (P773 = 0).

3.6 P775 - CAN controller state

This parameter gives information about the device state relating to the CAN bus. It indicates if the device is operating correctly or it informs which communication error type has been detected on the PLC2 board.

Range	*Default*	*Access*
0 = Communication disabled 1 = Not used 2 = No fault 3 = Warning 4 = Error Passive 5 = Bus Off 6 = Bus not powered	-	Read only

These errors are function of the number of invalid telegrams received or transmitted to the network, as described in section 1.1.4. For instance, the passive error status occurs when only one equipment is connected to the network sending telegrams in sequence, and no other equipment recognizes these telegrams. The bus off state can occur, for instance, when devices with different communication baud rates are connected to the same network, or due to installation problems, as the lack of termination resistors.

3.7 P776 - Number of received telegrams

This parameter operates as a cyclic counter. It is incremented every time a CAN telegram is received. It informs the operator if the device is communicating with the network correctly.

Range	*Default*	*Access*
0 ... 65535	-	Read only

3.8 P777 - Number of transmitted telegrams

Like the parameter P776, this parameter operates as a cyclic counter. It is incremented every time a CAN telegram is transmitted. It informs the operator if the PLC2 is communicating with the network correctly.

Range	*Default*	*Access*
0 ... 65535	-	Read only

3.9 P778 - Number of recorded errors

Cyclic counter that informs how many times the PLC2 board enters into the bus off state on the CAN network.

Range	*Default*	*Access*
0 ... 65535	-	Read only

Always the device is switched off, these counters (P776, P777 and P778) are reset to 0 (zero) and start counting again. When the value 65535 is exceeded, the counters are reset and start counting again.

3.10 P779 - CANopen Operation Mode

This parameter indicates what is the operation mode for the PLC2 board CANopen interface.

Range	*Default*	*Access*
0 = Slave 1 = Master	-	Read only

The operation mode is defined using the WSCAN software, integrated to the PLC2 programming software WLP. The PLC2 board is initially configured as slave for the CANopen network. But using the WSCAN, it is possible to configure it as a CANopen master, or return to slave configuration (disabling the option ”master”).

For the description of how to operate the PLC2 as a network master, please see the WSCAN user’s guide.

3.11 P780 - CANopen network state

This parameter indicates the board status relating to the CANopen network, informing if the protocol has been enabled and if the error control service is enabled (Node Guarding).

Range	*Default*	*Access*
0 = Protocol disabled 1 = Not used 2 = CANopen enabled 3 = Node guarding enabled 4 = Node guarding error	-	Read only

To enable the protocol, it is necessary to program P770 = 1 and supply 24 VDC through CAN interface. Once the protocol is enabled, the device is ready for the network communication. One important function of the device for detecting errors both at the slaves and at the master is the node guarding service. For enabling this function, refer to section 5.6.2.

3.12 P781 - CANopen node state

The PLC2 board operates as slave on the CANopen network, having a state machine that controls its communication behavior. This parameter indicates the current state of the device.

Range	*Default*	*Access*
0 = Initialization 4 = Stopped 5 = Operational 127 = Pre-operational	-	Read only

The value 0 (zero) also indicates the protocol is not enable or the interface is not powered.

4 Object dictionary

The object dictionary is a listing containing several equipment data which can be accessed via CANopen network. An object of this listing is identified by a 16 bits index, and the data exchange between the devices is performed based on these objects.

The document CiA DS 301 defines a minimum number of objects that each CANopen network slave must have. The objects available in this listing are grouped according to the type of function they execute. The objects are arranged in the dictionary as follows:

Index	***Objects***	***Description***
0001h - 035Fh	Data types definition	Used as reference for the data type supported by the system.
1000h - 1FFFh	Communication objects	Objects common to all CANopen devices. It contains general information about the equipment and also the data for the communication configuration.
2000h - 5FFFh	Manufacturer specific objects	In this range, every CANopen equipment manufacturer can define freely which data these objects will represent.
6000h - 9FFFh	Standardized device objects	This range is reserved for the objects describing the behavior of similar equipment, irrespective the manufacturers. The PLC2 does not use this object range.

Table 8: Groupings in the object dictionary

Not referenced indexes in this listing are reserved for further use.

4.1 Dictionary structure

The general structure of the object dictionary has following format:

Index	Object	Name	Type	Access

- *Index*: indicates the object index in the dictionary directly.
- *Object*: describes which information the index stores (simple variable, array, record, etc.).
- *Name*: contains the object name in order to facilitate its identification.
- *Type*: indicates the stored data type directly. For simple variables, this data type may be an Integer, a float, etc. For arrays, it indicates the data type contained in the array. For records, it indicates the record format according to the data type described in the first part of the object dictionary (indexes 0001h - 035Fh).
- *Access*: informs if the related object is accessible only for read (ro), for read and write (rw), only for write (wo), or if it is a constant (const).

For objects of array or record types, a sub-index is still required, buts this is not described in the dictionary structure.

4.2 Data types

The first part of the object dictionary (index 0001h - 035Fh) describes the data type that can be accessed in a CANopen network device. These data may basic, Integer and float types or compound types, formed by an input set as records and arrays. Please find below the object types used by the PLC2 board.

4.2.1 Basic data types

The basic supported data types are following:

- *Signed integers*: there are three types of signed integers supported by the PLC2: INTEGER8, INTERGER16 and INTEGER32 representing integers formed by data with 8, 16 and 32 bits, respectively. Integers with signal are calculated by using a 2's complement, and during the transmission, always the least significant byte is transmitted firstly with the CAN telegram.
- *Unsigned integers*: there are three types of unsigned integers supported by the PLC2: UNSIGNED8, UNSIGNED16 and UNSIGNED32 representing integers formed by data with 8, 16 and 32 bits, respectively. Also during the transmission, always the least significant byte is transmitted firstly.

4.2.2 Compound data types

New data types can be formed through groupings of basic types into listings (arrays - formed by only one type of data) and structures (records - formed by different types of data). In this case, each item of this type is identified by a sub-index. Please find below a listing of the compound types used by the PLC2 board.

- *PDO_COMM_PARAMETER*: this record defines the information required for the PDO configuration for the CANopen communication. Section 5.3 shows the content and how each field is used.

Sub-index	**Entry description**	**Type**
00h	Number of supported entries in this record	UNSIGNED8
01h	COB-ID	UNSIGNED32
02h	Transmission type	UNSIGNED8
03h	Inhibit time	UNSIGNED16
04h	Reserved	UNSIGNED8
05h	Event timer	UNSIGNED16

Table 9: Record for PDOs configuration

- *PDO_MAPPING*: this record defines how to map the data which should be transmitted by a PDO during the CANopen communication. For more details about the content and the form how each field is used, refer to section 5.3.

Sub-index	Entry description	Type
00h	Number of mapped objects in PDO	UNSIGNED8
01h	1st mapped object	UNSIGNED32
02h	2nd mapped object	UNSIGNED32
⋮	⋮	⋮
40h	64th mapped object	UNSIGNED32

Table 10: Record for data mapping of a PDO

- *SDO_PARAMETER*: this record defines the information for configuring a SDO for the CANopen communication. For more details on how each field is used, refer to section 5.2.

Sub-index	Entry description	Type
00h	Number of supported entries in this record	UNSIGNED8
01h	COB-ID client → server	UNSIGNED32
02h	COB-ID server → client	UNSIGNED32
03h	Node-ID for client/server	UNSIGNED8

Table 11: Record for SDO configuration

- *IDENTITY*: this record is used for describing the device type that is present on the network.

Sub-index	Entry description	Type
00h	Number of supported entries in this record	UNSIGNED8
01h	Vendor-ID	UNSIGNED32
02h	Product Code	UNSIGNED32
03h	Revision Number	UNSIGNED32
04h	Serial Number	UNSIGNED32

Table 12: Record for device identification

4.2.3 Extended data types

The PLC2 board does not support extended data types.

4.3 Communication profile objects

In the object dictionary, the indexes 1000h to 1FFFh correspond to the part which responsible for the configurations of the CANopen network communication. These objects are common to all devices, however only some are mandatory. Please find below an object listing in the range supported by the PLC2.

Index	*Object*	*Name*	*Type*	*Access*
1000h	VAR	device type	UNSIGNED32	ro
1001h	VAR	error register	UNSIGNED8	ro

Index	***Object***	***Name***	***Type***	***Access***
1003h	ARRAY	pre-defined error field	UNSIGNED32	ro
1005h	VAR	COB-ID SYNC	UNSIGNED32	rw
100Ch	VAR	guard time	UNSIGNED16	rw
100Dh	VAR	life time factor	UNSIGNED8	rw
1014h	VAR	COB-ID EMCY	UNSIGNED32	ro
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
1402h	RECORD	3rd receive PDO Parameter	PDO CommPar	rw
1403h	RECORD	4th 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
1602h	RECORD	3rd receive PDO mapping	PDO Mapping	rw
1603h	RECORD	4th 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
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

Table 13: PLC2 object list - Communication Profile

The other objects which are not shown in this listing are not used by the PLC2, or they are arranged in reserved ranges.

4.4 Manufacturer specific objects

The manufacturer can define freely in the indexes 2000h to 5FFFh which objects will be present, as well as establishing the type and the function of each object. In the case of the PLC2, in this object range has been arranged the parameter list and communication markers. Through these parameters and markers you can operate the PLC2 board by executing any function that has bee programmed for the device.

These parameters are available in the index 2000h on, and by adding the number of the parameter to this index you will obtain its location in the dictionary. The communication markers initiate from objet 3CE2h, and the following markers are available:

- *%RW Read Word*: Read marker (network → PLC2), 16 bits long.
- *%RB Read Byte*: Read marker (network → PLC2), 8 bits long.
- *%WW Write Word*: Write marker (PLC2 → network), 16 bits long.
- *%WB Write Byte*: Write marker (PLC2 → network), 8 bits long.

Table below shows how these parameters and markers are arranged in this object dictionary.

Index	***Object***	***Name***	***Type***	***Access***
22EEh	VAR	P750 - Firmware Version	UNSIGNED16	ro
22EFh	VAR	P751 - Scan cycle ($\times$ 100μs)	UNSIGNED16	ro
22F0h	VAR	P752 - Clears the retentive markers	UNSIGNED16	rw
22F1h	VAR	P753 - Loads factory default	UNSIGNED16	rw
22F2h	VAR	P754 - Position reference	UNSIGNED16	ro
⋮	⋮	⋮	⋮	⋮
2320h	VAR	P800 - User Parameter	UNSIGNED16	rw
2321h	VAR	P801 - User Parameter	UNSIGNED16	rw
2322h	VAR	P802 - User Parameter	UNSIGNED16	rw
⋮	⋮	⋮	⋮	⋮
2383h	VAR	P899 - User Parameter	UNSIGNED16	rw
3CE2h	VAR	%RW0 - Read Word	UNSIGNED16	ro
⋮	⋮	⋮	⋮	⋮
3D01h	VAR	%RW31 - Read Word	UNSIGNED16	ro
3D02h	VAR	%RB0 - Read Byte	UNSIGNED8	ro
⋮	⋮	⋮	⋮	⋮
3D21h	VAR	%RW31 - Read Byte	UNSIGNED8	ro
3D22h	VAR	%WW0 - Write Word	UNSIGNED16	rw
⋮	⋮	⋮	⋮	⋮
3D41h	VAR	%WW31 - Write Word	UNSIGNED16	rw
3D42h	VAR	%WB0 - Write Byte	UNSIGNED8	rw
⋮	⋮	⋮	⋮	⋮
3D61h	VAR	%WW31 - Write Byte	UNSIGNED8	rw

Table 14: PLC2 Object List - Manufacturer specific

For complete listing and detailed parameter description, refer to PLC2 board manual. Please consider that the inverter parameters to which the PLC2 is connected, can not be accessed directly through the CANopen network.

5 Communication objects description

This section describes with details the object indicated in table 13. It also describes the communication objects (COBs) operation referenced in section 1.7. For proper use of the available functions for the PLC2 board communication, you must be acquainted with the object operation.

5.1 Identification objects

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

5.1.1 Object 1000h - Device Type

This object provides a 32 bits code, describing the device and its functionality.

Index	1000h
Name	Device type
Object	VAR
Type	UNSIGNED32

Access	ro
PDO Mapping	No
Range	UNSIGNED32
Default value	0000.0000h

The code can be divided into two parts: 16 low-order bits describing the Profile type which is used by the device, and 16 high-order bits, indicating a specific function according to the specified Profile. The PLC2 board does not follow a Profile defined by the CAN specification, therefore this object shows 0 (zero).

5.1.2 Object 1001h - Error Register

This object indicates if an error is present on the device or not. The error type recorded for the PLC2 is as shown in table 15.

Index	1001h
Name	Error register
Object	VAR
Type	UNSIGNED8

Access	ro
PDO Mapping	Yes
Range	UNSIGNED8
Default value	0

Bit	***Meaning***
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication error
5	Reserved (always 0)
6	Reserved (always 0)
7	Manufacturer specific

Table 15: Structure of the Error Register

When the device shows some error, the equivalent bit must be activated. The first bit (generic error) must be activated with any error condition.

5.1.3 Object 1003h - Pre-defined error field

This objects stores the error list existing in the device and which have been reported via EMCY. The sub-index 0 (zero) indicates the number of occurred errors and the other sub-indexes inform the codes of the occurred errors. Once the error has been corrected, the code of the error is removed from the list. This list can be cleared by writing 0 (zero) into the sub-index 0 (zero). The PLC2 supports only one error code, thus the list has only one position.

Index	1003h
Name	Pre-defined error field
Object	Array
Type	UNSIGNED32

Sub-index	0
Description	Number of errors
Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	0
Sub-index	1
Description	Error code
Access	ro
PDO Mapping	No
Range	UNSIGNED32
Default value	0

The 32 bits error code is formed by two basic information: the number of the error according to the error table specified by the CiA and the number of the error occurred on the board, which are grouped as follows:

PLC2 error code	*CiA error code*
UNSIGNED16	UNSIGNED16

The specific PLC2 error codes are as indicated in table below. Also the CFW-09 errors, to which the PLC2 board is connected, are sent via network.

PLC2 error code	***CiA error code***	***Description***
50	8611h	Lag error
51	6300h	Fault during program saving
52	6200h	Two or more movements enabled simultaneously
53	6200h	Invalid movement data
54	6200h	Drive disabled
55	6200h	Incompatible program or out of memory limits
56	6300h	Wrong CRC
57	6200h	Shaft not referenced for the absolute positioning
59	8100h	Optional Fieldbus board is offline
60	8100h	Access error to the optional Fieldbus
65	8130h	Guarding error

Table 16: Error codes table

5.1.4 Object 1018h - Identity object

This section gives general information about the device.

Index	1018h
Name	Identity objetct
Object	Record
Type	Identity

Sub-index	0
Description	Number of entries
Access	ro
PDO Mapping	No
Range	UNSIGNED8
Default value	3
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	PLC2: 0000.0201h

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

Sub-index	4
Description	Serial number
Access	ro
PDO Mapping	No
Range	UNSIGNED32
Default value	-

The Vendor ID is a number that identifies the manufacturer at CiA. In this case, WEG Indústrias S.A. - Divisão Automação is represented by the number 0000.00123h. The product code is defined by the manufacture and it changes according to the board model. The revision number represents the firmware version installed in the equipment. The sub-index 4 represents a unique serial number for any WEG drive on CANopen network.

5.2 Service Data Objects - SDOs

The SDOs are responsible for the direct access to the object dictionary of a determined network device. They are used for configuration purpose and thus they have low priority, since they should not be used for the data communication required for the device operation.

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

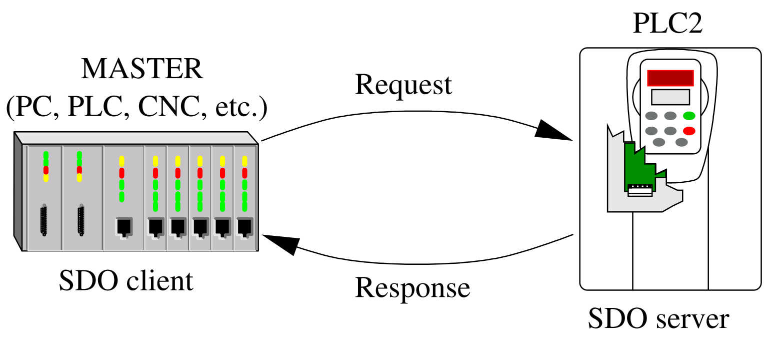


Figure 5: Communication between SDO Client and Server

5.2.1 Object 1200h - SDO Server

The PLC2 has an only SDO of server type that enables the access to the whole object dictionary. Through this dictionary, the SDO client can configure the device communication and parameters. Each SDO server has an object, of SDO_PARAMETER type (see section 4.2.2) for its configuration with the following structure:

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

Sub-index	0
Description	Number of entries
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

5.2.2 SDOs Operation

Every telegram sent by a SDO has an 8 bytes length and following format:

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

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

The command code depends on the used function type. Following commands can be used for the transmissions from the client to the server:

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 ou 70h	Upload segment	Segmented read	Not used

Table 17: Command codes for SDO client

Through the COB-ID the client must indicate in the request to which slave address the request is destined. Only a slave (using the respective SDO server) can answer the telegram to the client.

For readings that involve up to four data bytes, only one message can be transmitted by the server; for message readings with more bytes, the server must sent several telegrams. The response telegram has the same structure of the request telegram; however the commands will be different:

Command	***Function***	***Description***	***Object data***
60h	Download	Response to object write	Not used
43h	Upload	Response to object read	4 bytes
4Bh	Upload	Response to object read	2 bytes
4Fh	Upload	Response to object read	1 byte
41h	Upload segment	Begin of the segmented response for read	4 bytes
00h ou 10h	Upload segment	Data segment for read	4 bytes
06h ou 16h	Upload segment	Data segment for read	2 bytes
0Eh ou 1Eh	Upload segment	Data segment for read	1 bytes

Table 18: Command codes for SDO server

A telegram will be only completed after server has confirmed the client request. If during the telegram exchange any fault is detected (for instance, there is no server response), the client can abort the process by a message with command code equal to 80h.

NOTE!
When SDO is used for object write, representing the PLC2 parameters (object from index 2000h on), this value will be saved on the non-volatile board memory. Thus when equipment will be switched off or reset, the configured values will not be lost. These values will not be saved automatically for the other objects, thus the desired values must be re-written, or a command must be given for saving the values in the flash-memory by using the object 1010h.

Example: A SDO client requests the PLC2 board at address 1 to read the object identified by the index 22EEh, sub-index 0 (zero) which represents a 16 bits unsigned integer as equipment firmware version (P750). The master telegram has following format:

Identifier	*Command*	*Index*		*Sub-index*	*Data*			
601h	40h	EEh	22h	00h	00h	00h	00h	00h

The PLC2 answers the request, indicating that the value of the respective object is equal to 00A0h[3] (version 1.10).

Identifier	*Command*	*Index*		*Sub-index*	*Data*			
581h	4Bh	EEh	22h	00h	6Eh	00h	00h	00h

[3]Please remember that with any integer data type, the transference order will be executed from the least significant bytes to the most significant bytes.

5.3 Process Data Objects - PDOs

The PDOs are used for data transmission and data reception during the device operation, thus requiring safe and quick transmission and reception. Therefore the PDOs have higher priority than the SDOs.

In the PDOs, only the data are transmitted in the telegrams (indexes and sub-indexes are omitted), which enables a more efficient transmission with higher data volume in an only telegram. But for this purpose, you must configure previously what should be transmitted through the PDO so that even when the index and the sub-index are not indicated, the telegram content can be known.

There are two types of PDOs: the receive PDOs and the transmit PDOs. The transmit PDOs send the data to the network, whilst the receive PDOs read and process these data. This enables the communication between slaves of the CANopen network, requiring only the configuration of a slave for transmitting this information and the configuration of one or more slaves for receiving this information.

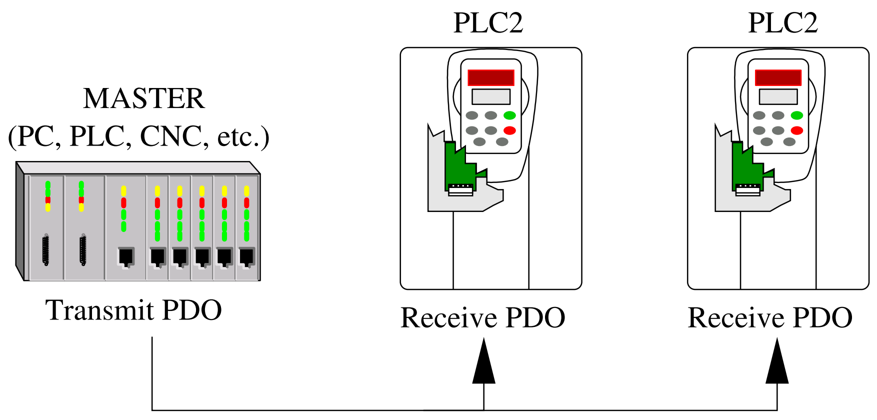


Figure 6: Communication by using PDOs

NOTE!
PDOs can only be transmitted or received when the device has been set to operational state. Fig. 9 shows the available states for a CANopen network node.

5.3.1 Mappable objects for the PDOs

To enable the transmission of an object through a PDO, it must be mappable to the PDO content. In the description of the communication objects (1000h - 1FFFh), the field PDO Mapping informs if the object is mappable or not. In general, the information required for the device operation is mappable, such as enable commands, devices status, reference, etc. Information about the device configuration can not be accessed through the PDOs. But when their access is required via network, you must use the SDOs.

For the specific PLC2 objects (2000h - 5FFFh), only the communication markers can be mapped in the PDOs. The markers %RW e %RB can be mapped to the RPDOs (objects 1600h to 1603h) whilst the markers %WW e %WB can be mapped to TPDOs (objects 1A00h to 1A03h). The content of these markers can be programmed via WLP software and thus the PLC2 board programming must be performed by considering which information will be used via CANopen network and at which parameters this information will be available.

Index	***Object***	***Name***	***Type***	***Access***	***PDO Mapping***
3CE2h	VAR	%RW0 - Read Word	UNSIGNED16	ro	RPDO
⋮	⋮	⋮	⋮	⋮	
3D01h	VAR	%RW31 - Read Word	UNSIGNED16	ro	RPDO
3D02h	VAR	%RB0 - Read Byte	UNSIGNED8	ro	RPDO
⋮	⋮	⋮	⋮	⋮	
3D21h	VAR	%RW31 - Read Byte	UNSIGNED8	ro	RPDO
3D22h	VAR	%WW0 - Write Word	UNSIGNED16	rw	TPDO
⋮	⋮	⋮	⋮	⋮	
3D41h	VAR	%WW31 - Write Word	UNSIGNED16	rw	TPDO
3D42h	VAR	%WB0 - Write Byte	UNSIGNED8	rw	TPDO
⋮	⋮	⋮	⋮	⋮	
3D61h	VAR	%WW31 - Write Byte	UNSIGNED8	rw	TPDO

Table 19: Markers list that can be mapped to PDOs

5.3.2 Receive PDOs

The Receive PDOs (RPDO) receive the data sent by other devices to the CANopen network. The PLC2 board has 4 Receive PDOs. One RPDO is able to receive up to 8 data bytes, and each RPDO has two configuration parameters: one PDO_COMM_PARAMETER and one PDO_MAPPING, as described below:

PDO_COMM_PARAMETER

Index	1400h - 1403h
Name	Receive PDO communication parameter
Object	Record
Type	PDO_COMM_PARAMETER

Sub-index	0
Description	Largest sub-index supported
Access	ro
PDO Mapping	No
Range	UNSIGNED8
Default value	2
Sub-index	1
Description	COB-ID used by PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1400h: 200h + Node-ID 1401h: 300h + Node-ID 1402h: 400h + Node-ID 1403h: 500h + Node-ID

Sub-index	2
Description	Transmittion type
Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	254

The sub-index 1 has the COB-ID of the RPDO. Always a message is sent to the network, this object will read which is the COB-ID of this 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 following structure:

Bit	***Value***	***Description***
31 (MSB)	0	PDO is enabled
	1	PDO is disabled
30	0	RTR allowed
29	0	Length of the identifier = 11 bits
28 - 11	0	Not used by the PLC2, always 0
10 - 0 (LSB)	X	COB-ID of 11 bits

Table 20: Description of PDO COB-ID entry

The bit 31 allows PDO enable or disable. The bits 30 and 29, which must be set to 0 (zero), indicate that the PDO accepts the remote frames (RTR frames) and that it uses an 11 bits identifier. As the PLC2 board does not use 29 bits identifiers, the bits 28 - 11 must be set to 0 (zero), whilst the bits 10 to 0 (zero) are used to configure the COB-ID to PDO.

The sub-index 2 indicates the type of transmission of these objects, as shown in table below.

Transmission type	***PDO transmission***				
	Cyclic	*Acyclic*	*Synchronous*	*Asynchronous*	*RTR only*
0		•	•		
1 - 240	•		•		
241 - 251	Not used				
252			•		•
253				•	•
254				•	
255				•	

Table 21: Description of transmission type

- *Values 0 - 240*: any RPDO programmed to this range operates in similar way. When a message is detected, it will receive these data, but the received values will not be updated before the next SYNC telegram has been detected.
- *Values 252 e 253*: these values are not allowed for Receive PDOs.
- *Values 254 e 255*: they indicated that there is no relationship with the synchronization object. When a message is received, they values are updated immediately.

PDO_MAPPING

Index	1600h - 1603h
Name	Receive PDO mapping
Object	Record
Type	PDO_MAPPING

Sub-index	0
Description	Number of mapped objects
Access	ro
PDO Mapping	No
Range	0 = disabled 1 ... 4 = number of mapped objects
Default value	1600h: 4 1601h: 4 1602h: 4 1603h: 4
Sub-index	1
Description	1st mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1600h: 3CE2.0010h 1601h: 3CE6.0010h 1602h: 3CEA.0010h 1603h: 3CEE.0010h
Sub-index	2
Description	2nd mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1600h: 3CE3.0010h 1601h: 3CE7.0010h 1602h: 3CEB.0010h 1603h: 3CEF.0010h
Sub-index	3
Description	3rd mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1600h: 3CE4.0010h 1601h: 3CE8.0010h 1602h: 3CEC.0010h 1603h: 3CF0.0010h

Sub-index	4
Description	4th mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1600h: 3CE5.0010h 1601h: 3CE9.0010h 1602h: 3CED.0010h 1603h: 3CF1.0010h

This object indicates the objects which have been mapped for each one of the four available receive PDOs. For each PDO can be mapped up to four different objects, provided the total length does not exceed eight bytes. The object mapping is executed by indicating it index, sub-index (If the object is VAR type and does not have sub-index, you must indicate 0 (zero) for the sub-index) and length (in bits) in the field UNSIGNED32 having following format:

UNSIGNED32		
Index (16 bits)	Sub-index (8 bits)	Object length (8 bits)

As example, we will analyze the first RPDO, where:

- *Sub-index 0 = 4*: RPDO has four mapped objects.
- *Sub-index 1 = 3CE2.0010h*: the first mapped object has an index equal to 3CE2h (corresponding to the marker %RW0), sub-index 0 (zero) and a length of 16 bits.
- *Sub-index 2 = 3CE3.0010h*: the second mapped object has an index equal to 3CE3h (corresponding to the marker %RW1), sub-index 0 (zero) and a length of 16 bits.
- *Sub-index 3 = 3CE4.0010h*: the third mapped object has an index equal to 3CE4h (corresponding to the marker %RW2), sub-index 0 (zero) and a length of 16 bits.
- *Sub-index 4 = 3CE5.0010h*: the fourth mapped object has an index equal to 3CE5h (corresponding to the marker %RW3), sub-index 0 (zero) and a length of 16 bits.

Therefore always this PDO receives a telegram, the PDO knows that this telegram must have 8 bytes of data, with the content of the markers %RW0 to %RW3 - these markers may be used by the user for the PLC2 board programming. These values are standard for the PLC2. You can change this mapping by changing the number of mapped parameters. Please consider that up to 4 objects or 8 bytes max. can be changed and that for the RPDOs you can map the markers shown in table 19).

NOTE!
• For changing the objects mapped in PDO, you must write firstly the value 0 (zero) in the sub-index 0 (zero). Now you can change the values of the sub-indexes 1 to 4. After ending the desired mapping, you must enter again in the sub-index 0 (zero) the number of the objects that have been mapped for enabling PDO again.
• For speeding up the data updating via PDO, the values received through these objects are not saved on the PLC2 non-volatile memory. So when the equipment is switched off or reset, the changed values will not be restored.

5.3.3 Transmit PDOs

The Transmit PDOs (TPDO) are responsible for the data transmission to the CANopen network. The PLC2 has 4 Transmit PDOs, each one containing up to 8 data bytes. Similar to the RPDOs, each TPDO has also two configuration parameters: one PDO_COMM_PARAMETER and one PDO_MAPPING, as described below:

PDO_COMM_PARAMETER

Index	1800h - 1803h
Name	Transmit PDO Parameter
Object	Record
Type	PDO_COMM_PARAMETER

Sub-index	0
Description	Largest sub-index supported
Access	ro
PDO Mapping	No
Range	UNSIGNED8
Default value	5
Sub-index	1
Description	COB-ID used by PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1800h: 180h + Node-ID 1801h: 280h + Node-ID 1802h: 380h + Node-ID 1803h: 480h + Node-ID
Sub-index	2
Description	Transmission type
Access	rw
PDO Mapping	No
Range	UNSIGNED8
Default value	254
Sub-index	3
Description	Inhibit time
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 = desabilitado UNSIGNED16
Default value	0

The sub-index 1 contains the COB-ID of the Transmit PDO. Always the TPDO sends a message to the network, the COB-ID will identify this message. Table 20 describes the structure of this field.

The sub-index 2 indicates the type of transmission of this object which is as indicated in table 21. However its operation is different for the transmission PDOs:

- *Value 0*: indicates that the transmission must be made immediately after a SYNC telegram has been received, however the transmission will not be periodically. Not used by PLC2.
- *Values 1 - 240*: the PDO must be transmitted periodically always when a telegram is detected (or in the 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 SYNC telegram has been received. The message transmission should be executed after the remote frame reception (RTR frame).
- *Value 253*: the PDO shall send a message as soon as a remote frame is received.
- *Values 254 e 255*: the object must be transmitted according to the timer programmed at the sub-index 5.

In the sub-index 3 you can program the minimum time (multiples of 100μs) that must elapse after a telegram transmission and before a new telegram can be sent to this PDO. The setting of 0 (zero) disables this function.

The sub-index 5 contains a value for a timer for sending a PDO automatically. Always a PDO is configured to asynchronous type, you can program this timer (multiples of 1ms) so the PDO will be sent periodically within the programmed time. The max. allowed value for this timer is 3276ms.

NOTE!
• Please ensure that the time programmed for this timer meets the used baud rate. Too short time (near the telegram transmission time) monopolizes the bus, causing undefined PDO transmission and prevents that other objects with less priority can be transmitted.
• Please consider that the PDOs can be transmitted or received only in operational state.

PDO_MAPPING

Index	1A00h - 1A03h
Name	Transmit PDO mapping
Object	Record
Type	PDO_MAPPING

Sub-index	0
Description	Number of mapped objects
Access	ro
PDO Mapping	No
Range	0 = disabled 1 ... 4 = number of mapped objects
Default value	1A00h: 4 1A01h: 4 1A02h: 4 1A03h: 4
Sub-index	1
Description	1st mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1A00h: 3D22.0010h 1A01h: 3D26.0010h 1A02h: 3D2A.0010h 1A03h: 3D2E.0010h
Sub-index	2
Description	2nd mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1A00h: 3D23.0010h 1A01h: 3D27.0010h 1A02h: 3D2B.0010h 1A03h: 3D2F.0010h
Sub-index	3
Description	3rd mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1A00h: 3D24.0010h 1A01h: 3D28.0010h 1A02h: 3D2C.0010h 1A03h: 3D30.0010h

Sub-index	4
Description	4th mapped object in PDO
Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	1A00h: 3D25.0010h 1A01h: 3D29.0010h 1A02h: 3D2D.0010h 1A03h: 3D31.0010h

The PDO_Mapping for transmission operates in similar way as required for the reception, however in this case are defined the data that should transmitted by the PDO. Each mapped object should be indicated in the list as shown below:

UNSIGNED32		
Index (16 bits)	Sub-index (8 bits)	Object length (8 bits)

As example we will analyze the first TPDO, where:

- *Sub-index 0 = 4*: PDO has four mapped objects.
- *Sub-index 1 = 3D22.0010h*: the first mapped object has an index equal to 3D22h (corresponding to the marker %WW0), sub-index 0 (zero) and a length of 16 bits.
- *Sub-index 2 = 3D23.0010h*: the second mapped object has an index equal to 3D23h (corresponding to the marker %WW1), sub-index 0 (zero) and a length of 16 bits.
- *Sub-index 3 = 3D24.0010h*: the third mapped object has an index equal to 3D24h (corresponding to the marker %WW2), sub-index 0 (zero) and a length of 16 bits.
- *Sub-index 4 = 3D25.0010h*: the fourth mapped object has an index equal to 3D25h (corresponding to the marker %WW3), sub-index 0 (zero) and a length of 16 bits.

Therefore always this PDO transmits its data, it prepares its telegram in length of 8 bytes of data by considering the values of the markers %WW0 to %WW3. These values are standard for the PLC2. You can change this mapping by changing the number of the mapped markers. Please consider that up to 4 objects or 8 bytes max. can be mapped and that for the TPDOs you can map the markers shown in table 19).

NOTE!
To change the objects mapped in a PDO, you must write firstly the value 0 (zero) in the sub-index 0 (zero). Now you can change the values of the sub-indexes 1 to 4. For enabling PDO again you must write in the sub-index 0 (zero) the number of the objects that have been mapped after completing the desired mapping.

5.4 Emergency Object - EMCY

The Emergency Object (EMCY) is used for displaying when an error is detected in the device. Always an error is detected on the PLC2 board, this object sends an emergency message to the network. This message can be interpreted by an EMCY consumer (usually the network master), that can adopt an action, as programmed for the application, for instance, disabling the other network devices and displaying the error.

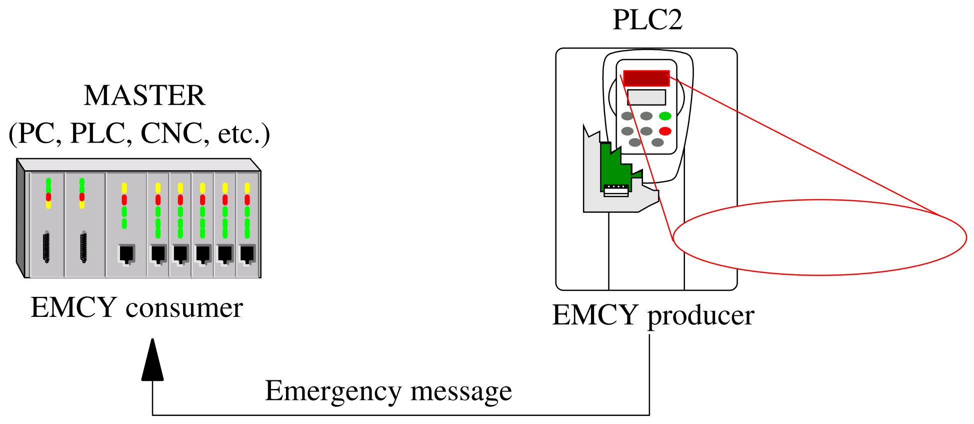


Figure 7: EMCY

When a message is transmitted, three fields are sent in the eight bytes of data: the CiA error code, the object 1001h (error register) and the PLC2 error code. These codes are described in table 16. The telegram has following structure:

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
CiA error code		Objeto 1001h Error Register	PLC2 error code		Not used (00h)		

There is only one parameter that can read the COB-ID of the object, i. e., which is the telegram identifier for the sent error messages.

Index	1014h
Name	COB-ID emergency message
Object	VAR
Type	UNSIGNED32

Access	ro
PDO Mapping	No
Range	UNSIGNED32
Default value	80h + Node-ID

5.5 Synchronization Object - SYNC

This object is transmitted for allowing the event synchronization between the CANopen network devices. It is transmitted by a SYNC producer and the devices that detect its transmission are called SYNC consumers.

The PLC2 has the SYNC consumer function and it can program its PDO as synchronous. As describes in table 21, synchronous PDOs are those related to the synchronization object an they can be programmed to be transmitted or updated by considering this object.

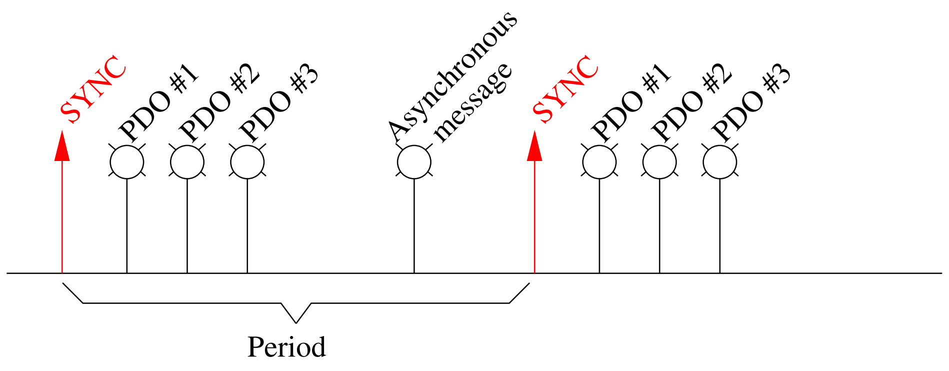


Figure 8: SYNC

NOTE!
Please consider that the time programmed in the producer for the SYNC telegrams periods must be set according to the used baud rate and number of the synchronous PDOs to be transmitted. Sufficient time must be provided for this purpose. Please provide also sufficient time for sending the asynchronous messages, as the EMCY, asynchronous PDOs and SDOs.

The SYNC message transmitted by the producer has a cleared data field, since its purpose is to provide a time basis for the other objects. There is an object on the PLC2 board for configuring the COB-ID of the SYNC consumer.

Index	1005h
Name	COB-ID SYNC
Object	VAR
Type	UNSIGNED32

Access	rw
PDO Mapping	No
Range	UNSIGNED32
Default value	80h

5.6 Network Management - NMT

The network management object provides several services for controlling the CANopen network devices. For the PLC2 board are available the node state control and the error control services (Node Guarding).

5.6.1 Slave state control

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

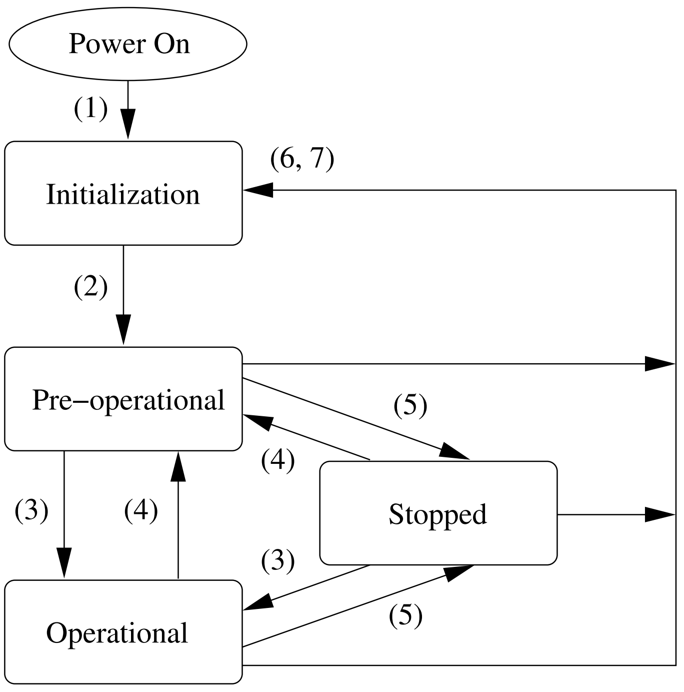


Figure 9: State diagram of the CANopen Node

Transition	***Description***
1	At power on the initialization state is entered autonomously
2	Initialization finished, enter Pre-operational state automatically
3	It receives the command Start Node to enter into the operational state
4	It receives the command Enter Pre-Operational and goes to the pre-operational state
5	It receives the command Stop Node and enters into the stopped state
6	It receives the command Reset Node and executes the complete device reset
7	It receives the command Reset Communication and reinitializes the objects value and the CANopen device communication.

Table 22: State transition description

During the initialization is defined the Node-ID, created the objects and configured the CAN network interface. During this stage you can not communicate with the device which is ended automatically.

At the end of this stage, the slave sends a Boot-up telegram to the network, which for indicating only that the boot has been concluded and that the slave entered into the pre-operational state. This telegram has the identifier 700h + Node-ID and only one byte of data with value equal to 0 (zero).

In the pre-operational state you can already communicate with the slave, however the PDOs are not available for operation yet. In the operational state all objects are available, whilst in the stopped state, only the object NMT can receive or transmit telegrams to the network. Table below shows the objects which are available in each state.

	Initialization	*Pre-operational*	*Operational*	*Stopped*
PDO			●	
SDO		●	●	
SYNC		●	●	
EMCY		●	●	
Boot-up	●			
NMT		●	●	●

Table 23: States and communication objects

As describes in section 3.12, the Parameter P781 allows checking the current PLC2 state.

The state machine is controlled by the network master that sends command to each slave for executing the desired state transition. These telegrams are not confirmed. This means that the slave receives the telegram only without answering to the master. The received telegrams have following structure.

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

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

Table 24: NMT Commands

The transitions indicated in the command code are similar to the state transition executed by the node after receiving the code (as shown in figure 9). The command Reset Node causes the PLC2 for executing a complete device reset, whilst the command Reset Communication causes the device for reinitializing only the objects relating to the CANopen communication.

5.6.2 Error control - Node Guarding

Two services are available for the device error control: Heartbeat and Guarding, but only the Guarding has been implemented in the PLC2.

The Guarding enables monitoring the communication with the CANopen network both by master and by slave. In this service type, the master sends telegrams periodically to the slave that responds to the received telegram. If any error is detected which interrupts the communication, you can identify this error, since both the master and the slave are notified by the timeout during the execution of this service. The error events are called Node Guarding by the master and as Life guarding by the slave.

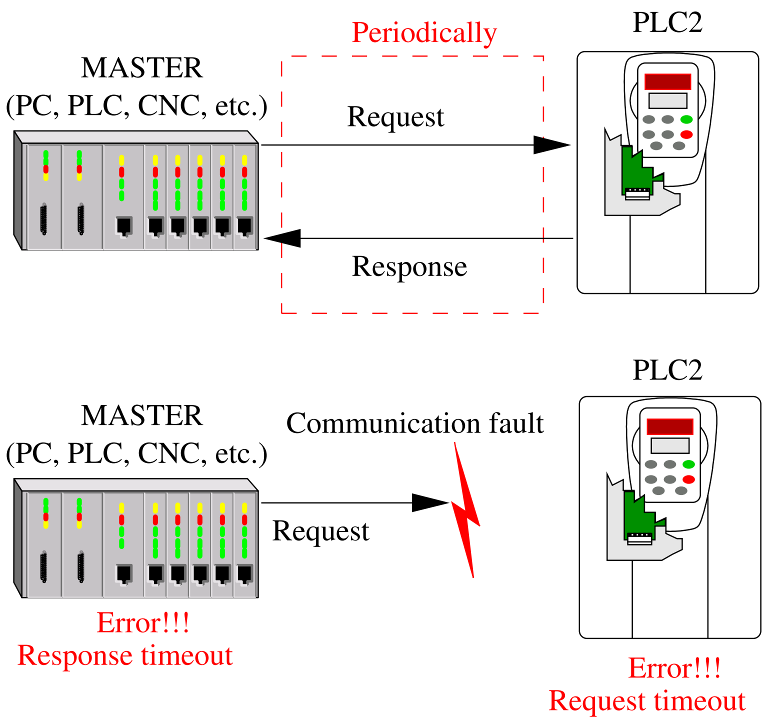


Figure 10: Error control service - Guarding

For the guarding service of the CANopen device, there are two objects in the dictionary for configuring the timer for the error display.

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 0 = Desabilitado
Default value	0

The object 100CH allows programming the required time (in milliseconds) for detecting an fault occurrence when PLC2 does not receive any guarding telegram from master. The object 100DH indicates how many sequence faults are required till a communication error will be considered. Thus the multiplication of these two values will provide the total time required for communication error detection by using this object. The value 0 (zero) will disable this function.

Once the PLC2 has been configured, it starts to count these times beginning with the first guarding telegram received from the master.

The master sends a remote frame that does not have data bytes. The identifier is equal to 700h + Node-ID of the destination slave. The slave response telegram has a data byte with following structure:

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

This byte contains, at its seven least significant bits, a value for indicating the slave state (4 = Stopped, 5 = Operational and 127 = Pre-operational and at its eighth bit a value that must be changed at every guarding telegram sent by the slave (toggle bit).

If the PLC2 detects an error by using this mechanism, it goes automatically to the pre-operational state. The board state on the network can be checked at parameter P781, whilst the occurrence of a guarding error can be checked at parameter P780.

Through the parameter P774 you can also program the PLC2 for adopting an action when this error is detected. For more details about this parameter, refer to section 3.

NOTE!
For using this service, please note following:
- This object is active even in stopped state (table 23).
- The max. allowed value for this timer is 3276ms.
- The enabling times of this function must be programmed both for the master and the slave on the network, so they can operate jointly.
- By considering the baud rate and the number of network points, the time must be so programmed that they are coherent and there is sufficient time for transmission of the guarding telegrams and the remaining communication can be processed.

5.7 Initialization procedure

After the operation of the available objects for the PLC2 board is known, you must now program the different objects for operating jointly on the network. In general, the object initialization procedures on a CANopen network are as shown in the flow chart below:

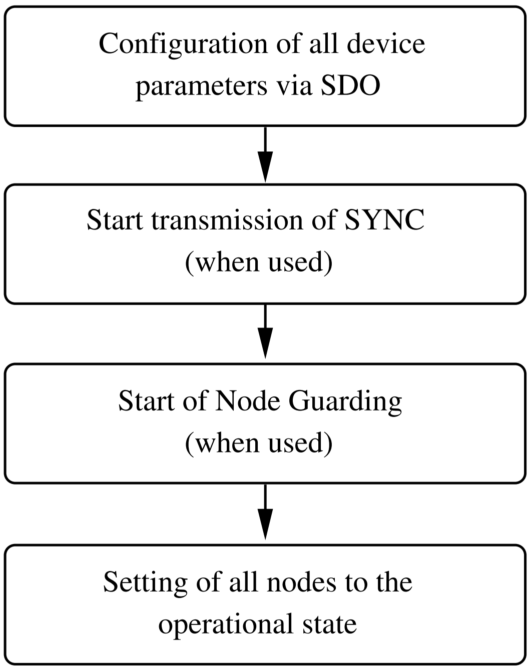

Configuration of all device parameters via SDO
Start transmission of SYNC (when used)
Start of Node Guarding (when used)
Setting of all nodes to the operational state

Figure 11: Network initialization process flow chart

Please note that the PLC2 communication objects (1000h to 1FFFh) are not stored automatically in the non-volatile board memory. So for each network device you must adopt following strategies:

- save the device configuration by using the object 1010h, or
- Always when equipment is reset or switched off, configure the communication objects again.

The specific manufacturer objects (2000h to 5FFFh), equivalent to the PLC2 board parameters, are stored automatically into the non-volatile memory, thus parameter setting is not required at every initialization.

6 CANopen communication errors

Please find below a description of the PLC2 specific errors for the CAN interface and the CANopen protocol.

6.1 E61 - *Bus off*

When a too high number of communication errors is detected by a CAN network device, this device can enter into the bus off state and then stopping the bus accessing. When this condition occurs with the PLC2, the drive keypad will display the error message E61, and it is necessary to reset the device in order to enable communication again. Please note that when the parameter P773 has been programmed for resetting the bus off automatically, this error will be disregarded and it will not be displayed.

This error can be caused due to several problems. Please consider following items for solving this problem:

- Communication baud rate has been set incorrectly. All CANopen network devices must be set to the same communication baud rate. If some device has been set incorrectly, this may cause an error in this device or in the other devices.
- Without termination resistor. Enable the termination resistors at the bus end to ensure trouble-free communication.
- Incorrect installation. Check all cables to ensure that no connection has been reversed, that all cables have been installed correctly and that the cable and the devices are grounded properly.

6.2 E63 - No power supply

When the CANopen protocol is enabled, the CAN interface must receive a voltage supply. If the CAN interface is not powered up, the product HMI will display the error message E63 and the CANopen communication will be disabled. The protocol will only operate again after the power supply has been restored.

6.3 E65 - Node Guarding error

One of the available services for the PLC2 is the error control using Guarding mechanism, where telegrams are exchanged periodically for ensuring that the communication is made without problems (Node Guarding). After the telegram exchange has started and the communication is interrupted during a time longer than the programmed one, the board will display on the drive keypad the error message E65, indicating that there is a Node Guarding error.

This function depends on the configurations which have been set by the network master. Eventual errors may occur due to communication problems or improper programming of the master or objects through which this function is enabled. Please refer to section 5.6.2 for more details about this function.

NOTA!
Once these errors have been shown on the frequency inverter, they will be erased only if user presses any key of the keypad. To have the update communication status, it is recommended to check the PLC2 status parameters, according to described at section 3.

7 Application examples

This section gives examples about parameter setting and operation procedures with a PLC2 board connected to a CFW-09 frequency inverter and operated via CANopen network. Following network configuration is considered in the examples:

- *1 Master*: a generic programmable equipment will be used with interface to the CANopen network having following functions:
  - Network master (controls the slave states and node guarding).
  - SYNC Producer
  - EMCY Consumer
  - SDO Client

  The PLC2 board can also be used for this function if configured as network master.

- *2 Slaves*: as CANopen network slaves will be used 2 PLC2 boards installed in 2 CFW-09.

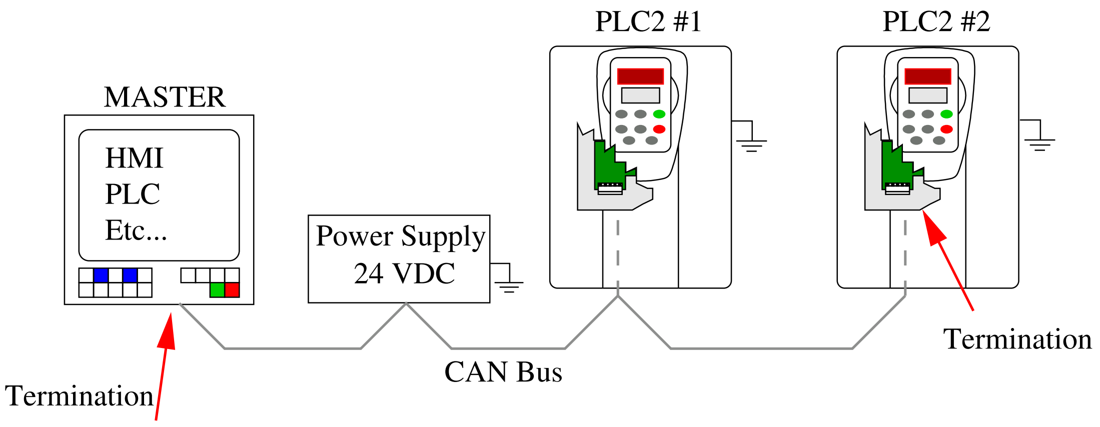


Figure 12: Network used in the application examples

Both CAN bus ends must be fitted with a 120Ω / 0.25W termination resistor connected to the signals CAN_H and CAN_L. When it is not provided by the device, this resistor can be fitted directly to the CAN connector. All network points must be grounded, and if possible, connected to the common grounding point. Use a power supply to supply the CAN interface via bus.

For the CANopen communication, following parameters are important for the PLC2 board parameter settings:

- P770: you must enable the CANopen protocol for both PLC2 boards. Thus you must set P770 to 1.
- P771: the Node-ID is set at P771. This setting must be different for each network slave. In our example, one PLC2 board will be programmed to the address #1 and the other to the address #2
- P772: considering that the network cable length will be shorter than 40 m, the max. baud rate will be used (1 Mbit/s). This is achieved by setting P772 to 0 (zero).

- P773: this parameter can be programmed according to the required application. In this example, the reset of bus off error must be executed in manual way (P773 = 0 (zero)).
- P774: if a communication error occurs with the PLC2 board on the CANopen network, this must cause a drive error. This is achieved by setting P774 = 1.

After parameter setting you must switch off/on or reset the drive to initialize the PLC2 board and so the new configurations could be accepted by the device. For diagnostic procedures you must observe following parameter sequence:

- P775: indicates the CAN network status. This parameter must display 2 (no error) when the CANopen protocol has been programmed, when the baud rate is correct and when the equipments are connected to the network.
- P780: this parameter must display 2 indicating that the CANopen protocol has been started without problems.
- P781: this parameter must display the value 127 while the network master does not start the PLC2 operation, indicating that the device is in pre-operational state.

After installation and parameter setting, the PLC2 will be now ready to be operated via CANopen network.

7.1 Example 1 - Slave state controlling

Section 5.6.1 shows a state diagram describing the behavior of a network slave. The slave state is controlled by the master through the managing services.

After initialization all slave are in pre-operational state. According to the sent telegram, the master can now request the reset or enter the slave into an existing state. Please find below some telegram examples. Please note that the telegrams are not confirmed by the master, i.e., the master send only messages to the network.

Master sends Start Node command to the slave at address 1

Master →	**000h**	**01h**	**01h**		→	Network

Master sends Stop Node command to the slave at address 1

Master →	**000h**	**02h**	**01h**		→	Network

Master sends Enter Pre-operational command to the slave at address 2

Master →	**000h**	**80h**	**02h**		→	Network

Master sends Reset Communication command to the slave at address 2

Master →	**000h**	**82h**	**02h**		→	Network

Master sends Reset Node command to all slaves on the network

Master	→	**000h**	**81h**	**00h**		→ Network

Please note that the COB-ID of the telegram is always 0 (zero). All slaves will receive the telegram, however only the slave with the destination Node-ID (byte 2) will execute the command (byte 1). If the byte 2 has the value 0 (zero), this means that the message is of broadcast type and that all slaves must execute the command.

7.2 Example 2 - SDOs messages

In this example, the telegrams will be shown via SDO, where the master reads and write the parameters and other network slave objects. Please note that the SDO telegram will have following structure:

COB-ID	*Command*	*Index*		*Sub-index*	*Object data*			
11 bits	byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7

Object read 1018h, sub-index 2 (Product code) of the address 1

Master	→	**601h**	**40h**	**18h**	**10h**	**02h**	00h	00h	00h	00h	→ PLC2
Master	←	**581h**	**43h**	**18h**	**10h**	**02h**	**01h**	**02h**	**00h**	**00h**	← PLC2

Response with the object content with value equal to 0000.201h

Write of P762 = 10 (object 22FAh) at address 2

Master	→	**602h**	**2Bh**	**FAh**	**22h**	**00h**	**0Ah**	**00h**	00h	00h	→ PLC2
Master	←	**582h**	**60h**	**FAh**	**22h**	**00h**	00h	00h	00h	00h	← PLC2

Response confirming parameter write

Read of P755 (object 22F3h) at address 2

Master	→	**602h**	**40h**	**F3h**	**22h**	**00h**	00h	00h	00h	00h	→ PLC2
Master	←	**582h**	**4Bh**	**F3h**	**22h**	**00h**	**E8h**	**03h**	00h	00h	← PLC2

Response with P755 content equal to $100,0^o$

7.3 Example 3 - Error control enabling - Guarding

The error control service allows the network integrants detecting errors which occur on remote devices. For this detecting service, the PLC2 board uses the Guarding service.

For configuring this service, you must proceed as follows:

- Which action should be adopted by the PLC2 board relating to CFW-09 when a network fault is detected?

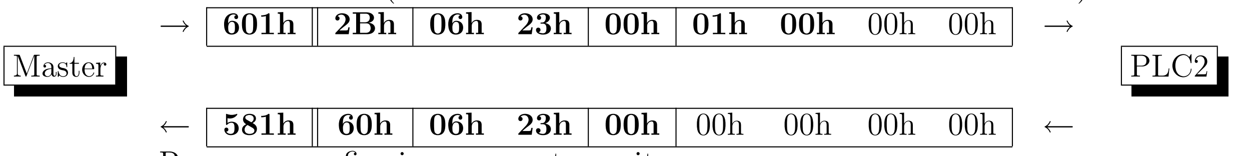
For this example, the PLC2 should cause a fatal error in the drive when a communication error is detected, disabling the motor. But for this purpose you must program the Parameter P774 to 1, as described in section 3.5. This setting can be made directly by the equipment HMI or through the CANopen network by using a SDO to write in the object 2306h (equivalent to P774).

- How many time can elapse for the PLC2 to detect a network fault?

  For this application we will consider that the limit time for detecting this communication error is 1 second. Based on this time, you can define the objects 100Ch - Guard time and 100Dh - Life time factor so the device will act within the programmed time. In this case, you can set the object 100Ch = 500 (which means that the PLC2 will receive a message from the master every 500 ms, if no message is received, this will be considered as service fault), and the object 100Dh = 2 (which means that when two faults are detected in sequence, this will be considered as an communication error). These values must be written by using the SDO of the PLC2.

These configurations are set to the slave at address 1, and the configuration of the slave at address 2 can be set in similar way.

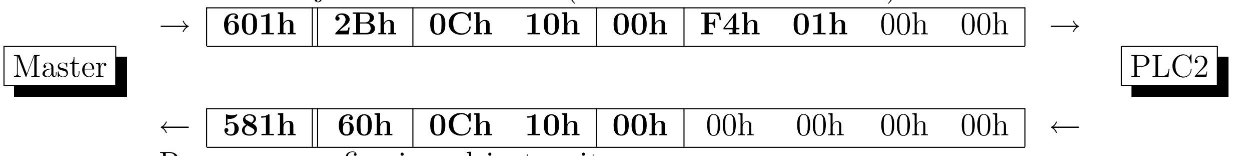
Write P774 = 1 (communication error causes faults in the CFW-09)



→ **601h**	**2Bh**	**06h**	**23h**	**00h**	**01h**	**00h**	00h	00h →
← **581h**	**60h**	**06h**	**23h**	**00h**	00h	00h	00h	00h ←

Response confirming parameter write

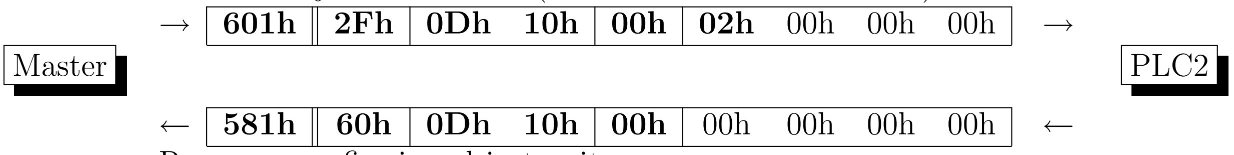
Write of object 100Ch = 500 (Guard Time = 500 ms)



→ **601h**	**2Bh**	**0Ch**	**10h**	**00h**	**F4h**	**01h**	00h	00h →
← **581h**	**60h**	**0Ch**	**10h**	**00h**	00h	00h	00h	00h ←

Response confirming object write

Write of object 100Dh = 2 (Life Time Factor = 2 times)



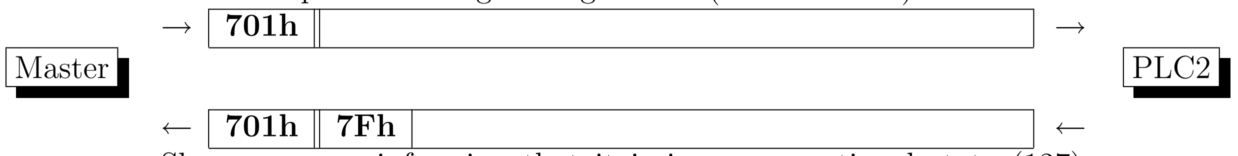
→ **601h**	**2Fh**	**0Dh**	**10h**	**00h**	**02h**	00h	00h	00h →
← **581h**	**60h**	**0Dh**	**10h**	**00h**	00h	00h	00h	00h ←

Response confirming object write

After these objects have been configured, the PLC2 board is ready for starting the Guarding service. The same service must be programmed in the Master[4], so the devices can operate jointly. Once the communication through the master has been started, the PLC2 starts the time counting as set for the error control.

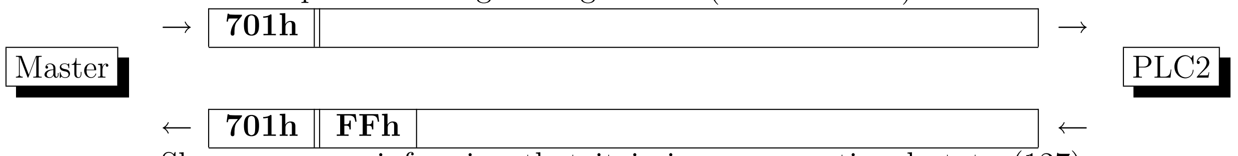
[4]The master is usually programmed with slightly shorter sending times in order to prevent timeout problems due to small time differences or delays for telegram sending.

Master request for the guarding service (remote frame)



Slave response informing that it is in pre-operational state (127). Toggle bit is disabled (bit 8)

Master request for the guarding service (remote frame)



Slave response informing that it is in pre-operational state (127). Toggle bit is enabled (bit 8)

The telegrams are sent cyclically. Once this service has been enabled, you can check its status through the Parameter P780 of the PLC2 board, which must be set to 3 (Node Guarding enabled).

When the communication is interrupted, this will cause an error in the CFW-09 and it goes to the pre-operational state (if it is not already in this state). The parameter P780 will assume the value 4, indicating that guard message exchange has been interrupted during a time longer than programmed in the objects 100Ch and 100Dh.

7.4 Example 4 - Configuring a Transmit PDO

Use the transmit PDOs for sending data from a determined equipment to the network. For the PLC2, the information that can be sent is in the markers %WW e %WB. These markers are programmable through the user program so the required data can be transmitted via CANopen network.

The information that should be transmitted must be mapped in one of the four transmit PDOs available for the PLC2. There is already a standard mapping, however, depending on the application, other information may be added or removed. Please note the following subjects:

- Which information shall be transmitted by the PLC2 to the network?

  You must define which information should be sent by the PDO. In our example, following information will be transmitted: effective position (rotation) and effective position (fraction of revolution) of the slave at address 1.

- In which parameters or marker this information is available?

  The effective position information is available at the parameters P757 and P758 of the board. So this information is available for the CANopen network, you must transfer the content of these parameters to the user parameters which can be mapped in the PDOs, as shown in the table 19. In this case, these parameters will be transferred to the markers %WW0 and %WW1 that are already mapped in the first transmit PDO. For transferring the content of these markers, you must program these markers through the software WLP.

- Which is the COB-ID for the message?

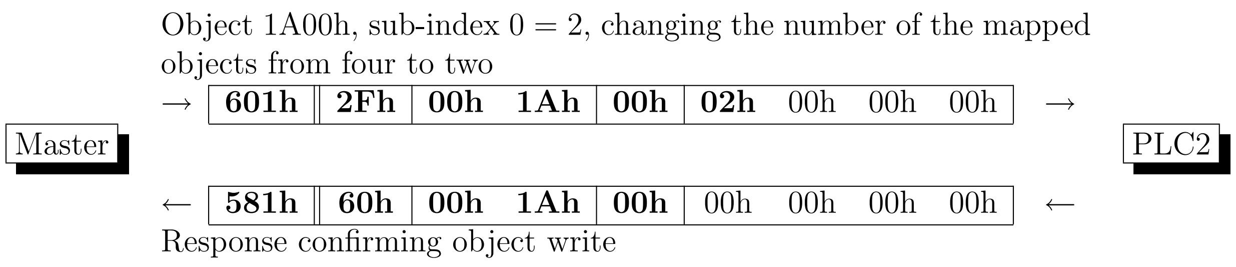
  When the PLC2 transmits this PDO, one or more equipment will receive this information. Thus it is important to know the COB-ID of the transmitted message so you can program the receive PDOs which will consume this message. In our example, the first transmit PDO will be used for the PLC2 at address 1. The COB-ID for the PDO will not be changed and the standard value (=181h) will be maintained (object 1800h, sub-index 1).

- How should this PDO be transmitted and how long is the time for this transmission?

  The PLC2 can transmit a PDO in several ways: linking this PDO to a telegram SYNC, programming the PDO timer so it will be transmitted periodically or by sending remote telegrams, requesting their transmission, as shown in table 21. In this case, the timer is used for the automatic transmission of the PDO at every 10 ms. The type of transmission is programmed in the object PDO_COMM_PARAMETER of the respective PDO, in the sub-index 5.

As described above, you must program firstly the software WLP for transferring the contents of the parameters P757 and P758 to the markers %WW0 and %WW1. For more detail about this programming, refer to the Programming Manual of PLC2 board.

The first transmit PDO of the board has already the mapped markers %WW0 and %WW1 (according to section 5.3.3). However there are other mapped objects which are not required, thus you must restrict the number of the mapped objects from four to only two objects.

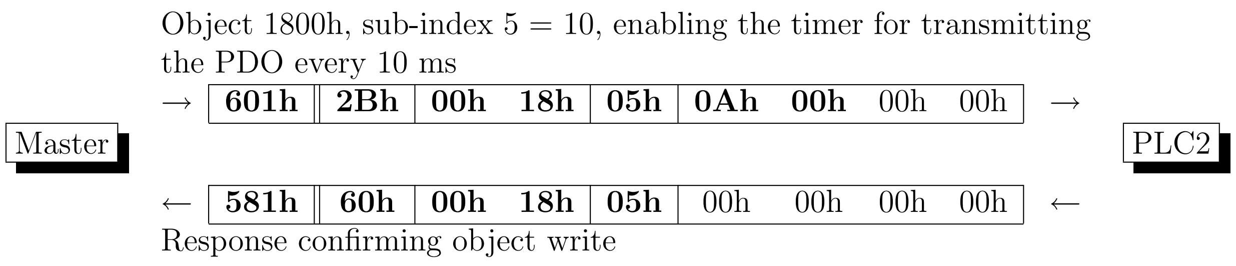


Object 1A00h, sub-index 0 = 2, changing the number of the mapped objects from four to two

→	**601h**	**2Fh**	**00h 1Ah**	**00h**	**02h**	00h	00h	00h	→
←	**581h**	**60h**	**00h 1Ah**	**00h**	00h	00h	00h	00h	←

Response confirming object write

Now you must configure the timer of the PDO so it is transmitted every 10 ms. This configuration is made in the object 1800h, sub-index 5. The values of the sub-indexes 2 and 3 will be maintained in their standard values.



Object 1800h, sub-index 5 = 10, enabling the timer for transmitting the PDO every 10 ms

→	**601h**	**2Bh**	**00h 18h**	**05h**	**0Ah**	**00h**	00h	00h	→
←	**581h**	**60h**	**00h 18h**	**05h**	00h	00h	00h	00h	←

Response confirming object write

Now the PLC2 is ready for starting the transmission of its first PDO, containing the information about the motor position. But this transmission will be started only after the master has sent the command to the slave to enter into the operational state, since this is the only state where the slave can transmit or receive PDOs.

Master sends command Start Node to the slave at address 1

Master → | **000h** | **01h** | **01h** | | → PLC2

PLC2 sends the PDO containing %WW0 = 0 and %WW1 = 0 (position of the motor shaft)

PLC2 → | **181h** | **00h** **00h** | **00h** **00h** | | → Network

PLC2 sends the PDO containing %WW0 = 0 and %WW1 = 13 (new position of the motor shaft)

PLC2 → | **181h** | **00h** **00h** | **0Dh** **00h** | | → Network

Now the information will be sent to the network at every 10 ms. If one or more devices of the network should receive these data, you must program reception PDO with the COB-ID of this message and performing the mapping as required.

7.5 Example 5 - Configuring a Receive PDO

The receive PDOs are mainly used for the reception of the control data and references for the device operation. Similar to the transmit PDOs, you can map the markers %RW and %RB in a RPDO which will be responsible for receiving the network data and storing them into the mapped parameters. As the configuration of a receive PDO depends on the application, you must provide following information:

- Which information should be received by the PLC2 from the network?

  You must define which information will be received by the PDO. In our example, the PLC2 at address 2 will be used for receiving an only user parameter that will be used as frequency inverter speed reference.

- In which parameters this information will be available?

  In our case, we chose the marker %RW0 for receiving this speed reference information. Please note that %RW0 is a programmable marker, and to use it as speed reference, you must program the PLC2 board with the software WLP. For more details about the programming, refer to the Programming Manual of the PLC2 board.

- Which is the COB-ID for this message?

  The first receive PDO of the PLC2 board will be used, and its standard COB-ID (=202h) will be maintained for this RPDO (object 1400h, sub-index 1). This PDO will receive only the network messages which have the same COB-ID, so another network element must send messages with this COB-ID.

- How is the reception of this PDO?

  As shown in table 21, a receive PDO may be linked to a SYNC telegram or not. In our example, the transmission type (object 1400h, sub-index 2) for the RPDO with its standard value (=254) will be maintained, indicating that there is no relationship with the SYNC telegrams. So always a telegram is received, the mapped object values will be updated automatically.

Based on this information, as first step you must map the chosen data. As the first receive PDO will be used, its mapping has already the marker %RW0 configured to the first position (object 1600, sub-index 1). This PDO has other not used mapped objects that will be excluded.

Object 1600h, sub-index 0 = 1, for indicating that there is only 1 mapped object in this PDO

Master → PLC2

602h	**2Fh**	**00h**	**16h**	**00h**	**01h**	00h	00h	00h

PLC2 → Master

582h	**60h**	**00h**	**16h**	**00h**	00h	00h	00h	00h

Response confirming object write

After mapping, you must program the PLC2 to the operational state. In this state all telegrams with COB-ID equal to 202h will be received by the device and the values of the mapped parameters will be updated.

Master sends command Start Node to the slave at address 2

Master → PLC2

000h	**01h**	**02h**	

PLC2 receives from the network the PDO containing %RW0 = 1000

Network → PLC2

202h	**E8h 03h**	

The received values are stored in the mapped parameters and they will be interpreted according to the program set for the PLC2 board by using the WLP.

7.6 Example 6 - Using the SYNC object for sending PDOs

The SYNC telegrams can be used as time base for transmission or reception of the network data. To use it with the PLC2 board, some network device must be the producer of this object. The PLC2 has an entry in its dictionary (object 1005h), where you can configure the COB-ID used by the SYNC producer.

Firstly you must define following subjects:

- Which time shall be used by the SYNC producer for the telegram transmission?

  In our example, the transmission time of a SYNC telegram will be 100 ms. This programming is only performed in the SYNC producer, which is in this case the master. The COB-ID for this telegram will be 80h, which is the standard value.

- Is there any transmit or receive PDO that should be linked to this object? For the transmit PDOs, they must be transmitted at every SYNC, or at SYNC multiple values.

  Each device will have a transmit PDO linked to the SYNC telegram. The board at the address 1 shall send its PDO at every detected SYNC telegram, whereas the board at the address 2 shall send its telegram at every 2 SYNC telegrams. The mapped data in the PDOs will be maintained with their standard values for the PLC2.

Now the PDOs must be configured by using the SDO of each slave.

Slave 1, object 1800, sub-index 2 = 1, programming the PDO to be transmitted with every SYNC telegram

Master	→	**601h**	**2Fh**	**00h**	**18h**	**02h**	**01h**	00h	00h	00h	→	PLC2
	←	**581h**	**60h**	**00h**	**18h**	**02h**	00h	00h	00h	00h	←	

Response confirming the object write

Slave 2, object 1800, sub-index 2 = 2, programming the PDO to be transmitted at every 2 SYNC telegrams

Master	→	**602h**	**2Fh**	**00h**	**18h**	**02h**	**02h**	00h	00h	00h	→	PLC2
	←	**582h**	**60h**	**00h**	**18h**	**02h**	00h	00h	00h	00h	←	

Response confirming the object write

After configuration has been completed, you must enable the production of SYNC telegrams in the master, according to the time that has been stipulated for the production (in our case, at every 100 ms). However the slaves will not transmit their PDOs before they do not enter into the operational status. Also the master will give this command.

Master starts the transmission of the SYNC telegrams at every 100 ms

Master	→	**080h**	→	PLC2

Master sends Start Node command to all slaves

Master	→	**000h**	**01h**	**00h**	→	PLC2

When the slaves enter into the operational state, the slave at the address 1 will send a PDO at every SYNC telegram, whereas the slave at address 2 will send a PDO at every 2 received SYNC telegrams. These telegrams will be repeated indefinitely during all time the system is operating. The data transmitted by the PDOs can be consumed by other devices on the network according to the desired operation logic.

Master sends a new SYNC telegram

Master	→	**080h**									→	PLC2

The PLC2 at address 1 sends a PDO with the content of the markers %RW0, %RW1, %RW2 and %RW3 (equal to zero)

PLC2	→	**181h**	**00h**	**00h**	**00h**	**00h**	**00h**	**00h**	**00h**	**00h**	→	Bus

Master sends a new SYNC telegram

Master	→	**080h**									→	PLC2

The PLC2 at address 1 sends a PDO with the content of the markers %RW0, %RW1, %RW2 and %RW3 (equal to zero)

PLC2	→	**181h**	**00h**	**00h**	**00h**	**00h**	**00h**	**00h**	**00h**	**00h**	→	Bus

The PLC2 at address 2 sends a PDO with the content of the markers %RW0 = 100, %RW1 = 200, %RW2 = 300 and %RW3 = 400 (values only for demonstration)

PLC2	→	**182h**	**64h**	**00h**	**C8h**	**00h**	**2Ch**	**01h**	**90h**	**01h**	→	Bus

In the same way as the SYNC telegram is used by the transmit PDOs, they can be also used by the receive PDOs for synchronizing the data reception by the network devices. When you program a RPDO to synchronous type, all data received by this PDO will be updated in the object dictionary only when the next SYNC telegram has been detected.

7.7 Example 7 - Error detection by using EMCY

The EMCY telegrams can be used for displaying any error that has been detected on the network device. The PLC2 has an EMCY producer that sends a message to the network always an error is detected and indicates the error type that has been detected. If the network master is fitted with an EMCY consumer which is monitoring this event, it can detect and display this error or adopt an action as programmed for this application.

The PLC2 boards have a single object for reading the COB-ID of this object (object 1014h), which as standard assumes the value 080h + Node-ID of the slave. Please find below some telegram examples sent by this object for displaying the detected error. The used error codes are shown in table 16.

The slave at address 1 sends an EMCY telegram, displaying lag error

PLC2	→	**081h**	**11h**	**86h**	**01h**	**32h**	**00h**	00h	00h	00h	→	Network

The slave at address 2 sends an EMCY telegram, displaying life guard error

PLC2	→	**082h**	**30h**	**81h**	**11h**	**41h**	**00h**	00h	00h	00h	→	Network