

Chapter 7 - Settings and Commissioning

7.1 PC Configurator

The *E@syDrivesPX* configurator is a program supplied together with the product. Its installation requires a PC with MS Windows ® 95/98/ME/XP or Windows NT®4/2000 system, with minimum 8 Mb RAM.

The configurator communicates with the drive using the Slink-3 protocol. Together with the drive parameterization, the configurator allows downloading the firmware in order to create some personalized applications using the MDPlc development environment.

7.2 Commissioning

Before powering up the drive, carry out the following verifications:

- Check the connections with the line L1, L2, L3
- Check the connections with the motor U, V, W (T1, T2, T3)
- Check the braking resistor connection (if present)
- Check the connections between the resolver or encoder and XE connector
- Check the input connection 24Vdc (if present)
- Check the I/O connections
- Check all the drive and motor ground connections

After having checked as shown above, it's possible now to power the drive; then check:

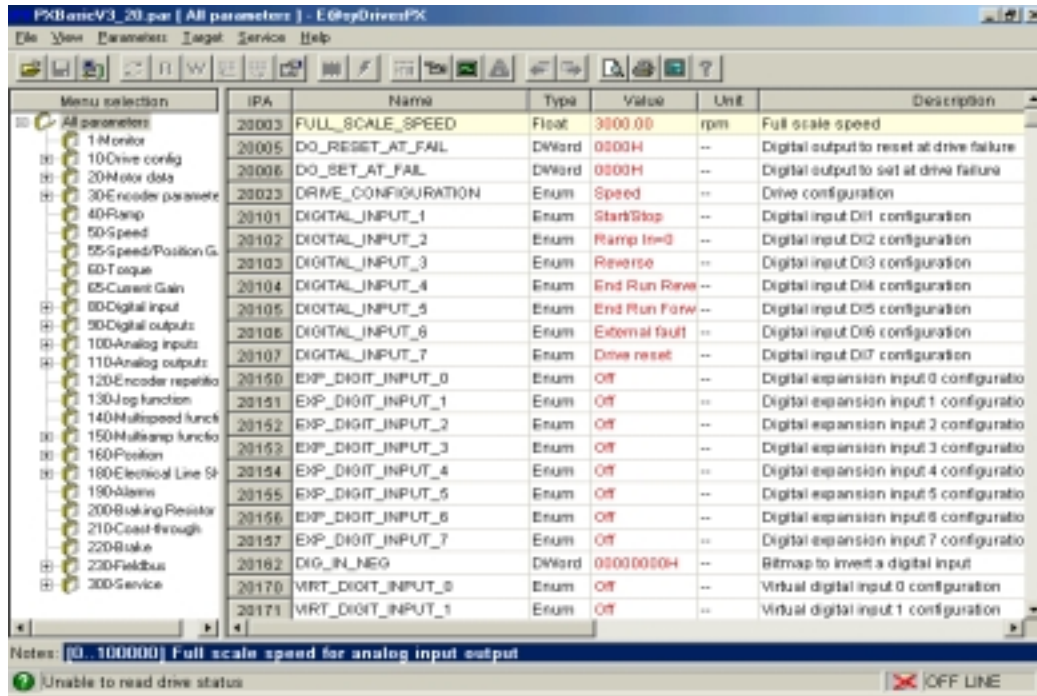
- Line voltage (max permissible voltage 480Vac + 10%)
- Voltage of the intermediate circuit DC bus (270-350Vdc for input voltage 230Vac, 432-528Vdc for input voltage 400Vac, 480-650Vdc for input voltage 480Vac). If the measured voltage is not in the indicated range, check the line voltage.

7.2.1 Connection with the PC

The drive is delivered from the factory with a standard configuration in the speed mode, unless otherwise specified. The input and output state is already programmed as in the following example, therefore user is able to start up the drive control and run the motor immediately (when the drive is ordered with a Powertec motor).

To perform the correct parameter settings, it is beneficial to use the *E@syDrivesPX* configurator. Install the software onto your PC following the instructions included with the software. Connect the drive to your PC using the serial communication as suggested in the manual (Sec. 4.4); check that the termination resistance switch is on the 120 ohm position.

Start up the configurator by clicking on the installed desktop icon. After starting the configurator, open from the File/Open menu the PxBasicVx_xxx.par file, where x_xxx states the version of the Basic firmware. This file includes the list of all the parameters resident in the drive. The data is split into several windows and a menu tree, typical of the Windows system, therefore easy to understand.



When the parameter file has opened, the PC will connected automatically with the drive and communicate. If you see no errors after opening the parameter file, the drive is communicating with the PC. (For commands and configurator specifications refer to the *E@syDrivesPX* instruction manual.)

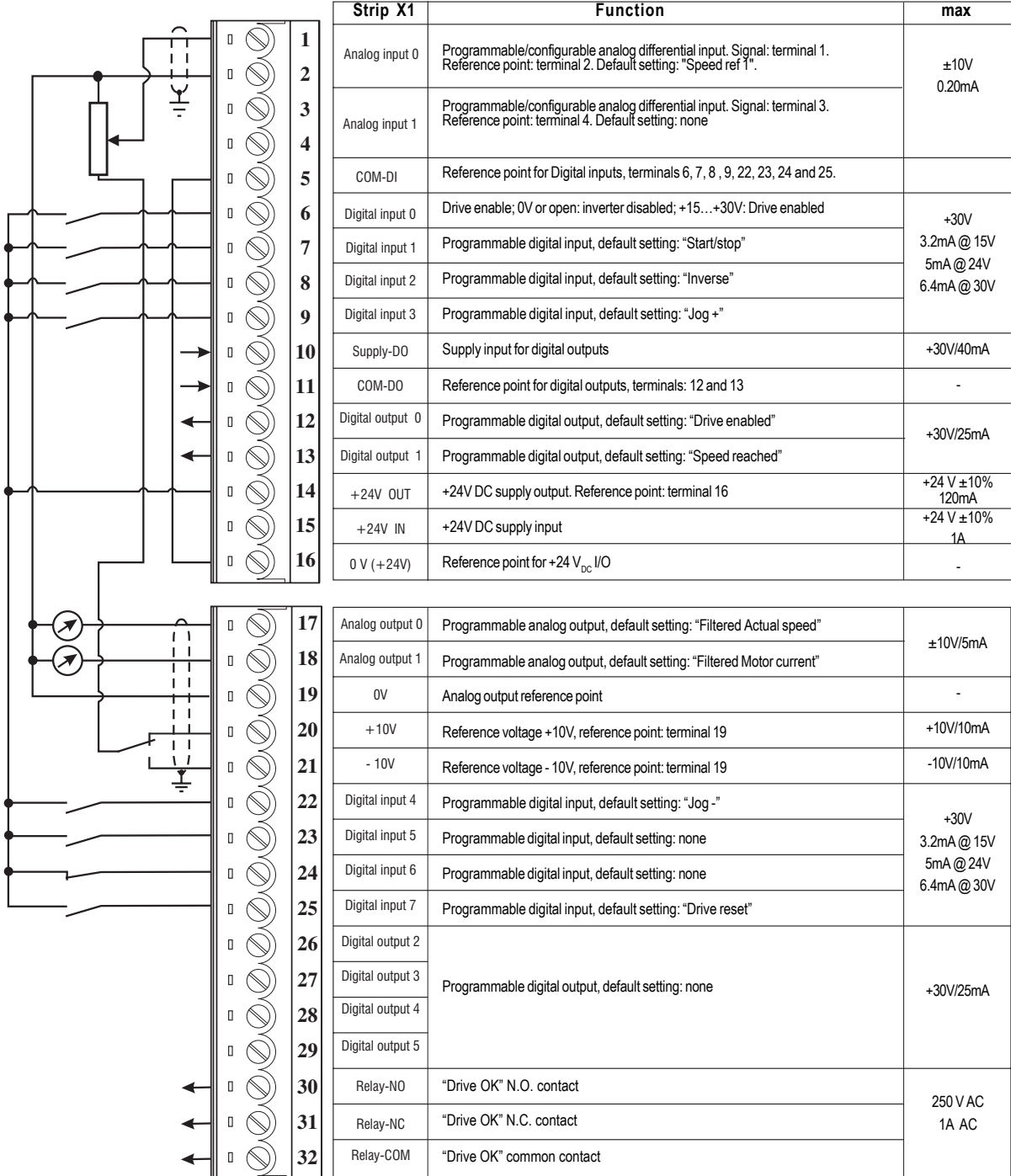
7.2.2 Essential Parameters Set up

The essential parameters to check before starting the motor are:

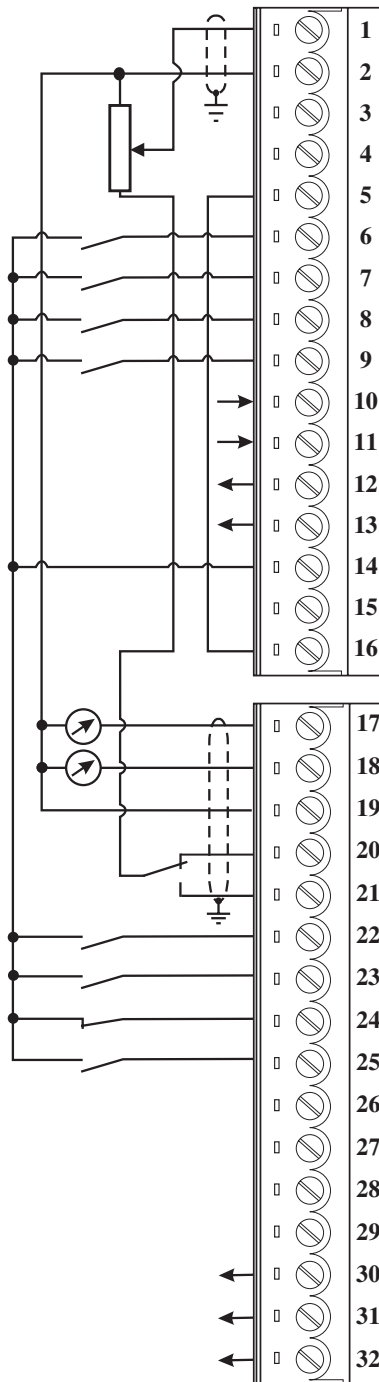
MENU	PARAMETER
001 - Monitor	20053 SYS_DRIVE_SIZE
010 - Drive Config	20000 SYS_I_MAX
020 - Motor Data	20002 SYS_MOT_NUMBER_OF_POLES
030 - Encoder parameter	20010 SYS_MAIN_ENC_TYPE
	20011 SYS_MAIN_ENC_PUL_REV
	20012 SYS_MAIN_ENCODER_SUPPLY
050 - Speed	20003 FULL_SCALE_SPEED

It's now possible to enable the drive and rotate the motor in the function of the inputs configuration and setup. As an example three types of configurations are described.

7.2.3 Speed Mode Configuration Example

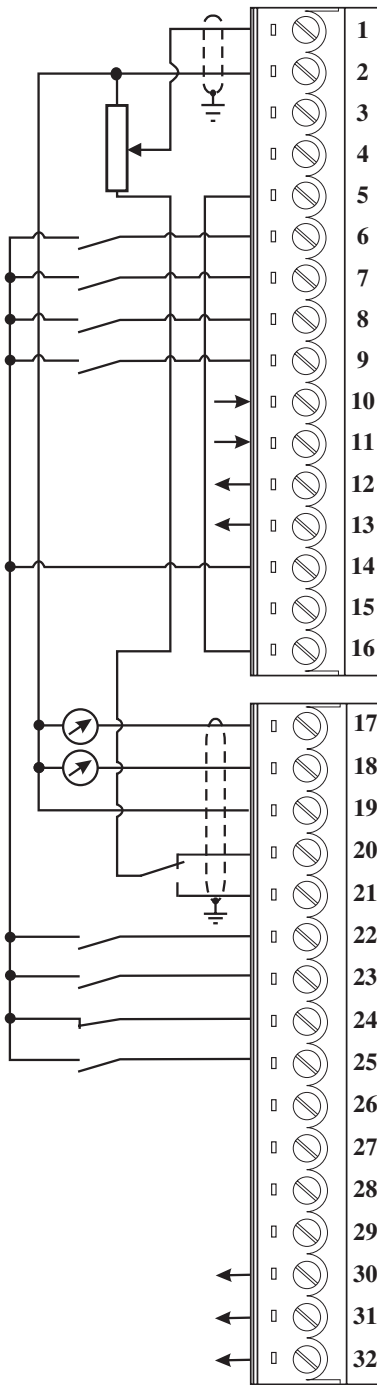


7.2.4 Position Mode Configuration Example



Strip X1	Function	max
1	Analog input 0	±10V 0.20mA
2	Not configured	
3	Analog input 1	Not configured
4	Not configured	
5	COM-DI	Reference point for Digital inputs, terminals 6, 7, 8, 9, 22, 23, 24 and 25.
6	Digital input 0	Drive enable; 0V or open: inverter disabled; +15...+30V: Drive enabled
7	Digital input 1	Programmable digital input, configured as: "Pos_Start Pos"
8	Digital input 2	Programmable digital input, configured as: "Pos_0 Search"
9	Digital input 3	Programmable digital input, configured as: "Pos_0 Sensor"
10	Supply-DO	Supply input for digital outputs
11	COM-DO	Reference point for digital outputs, terminals: 12 and 13
12	Digital output 0	Programmable digital output, default setting: "Drive enabled"
13	Digital output 1	Programmable digital output, configured as: "Pos_Pos reached"
14	+24V OUT	+24V DC supply output. Reference point: terminal 16
15	+24V IN	+24V DC supply input
16	0 V (+24V)	Reference point for +24 V _{DC} I/O
17	Analog output 0	Programmable analog output, default setting: "Filtered Actual speed"
18	Analog output 1	Programmable analog output, default setting: "Filtered Motor current"
19	0V	Analog output reference point
20	+10V	Reference voltage +10V, reference point: terminal 19
21	-10V	Reference voltage -10V, reference point: terminal 19
22	Digital input 4	Programmable digital input, default setting: "Jog -"
23	Digital input 5	Programmable digital input, default setting: none
24	Digital input 6	Programmable digital input, default setting: none
25	Digital input 7	Programmable digital input, default setting: "Drive reset"
26	Digital output 2	Programmable digital output, default setting: none
27	Digital output 3	
28	Digital output 4	
29	Digital output 5	
30	Relay-N0	"Drive OK" N.O. contact
31	Relay-NC	"Drive OK" N.C. contact
32	Relay-COM	"Drive OK" common contact

7.2.5 Electrical line Shaft Mode Configuration Example



Strip X1	Function	max
1	Analog input 0	±10V 0.20mA
2	Not configured	
3	Analog input 1	Not configured
4	Not configured	
5	COM-DI	Reference point for Digital inputs, terminals 6, 7, 8, 9, 22, 23, 24 and 25.
6	Digital input 0	Drive enable; 0V or open: inverter disabled; +15...+30V: Drive enabled
7	Digital input 1	Programmable digital input, configured as: "Start stop"
8	Digital input 2	Programmable digital input, configured as: "Els-Ratio Sel Bit 0"
9	Digital input 3	Programmable digital input, configured as: "Els-Ratio Sel Bit 1"
10	Supply-DO	Supply input for digital outputs
11	COM-DO	Reference point for digital outputs, terminals: 12 and 13
12	Digital output 0	Programmable digital output, configured as: "Current Limit"
13	Digital output 1	Programmable digital output, not configured
14	+24V OUT	+24V DC supply output. Reference point: terminal 16
15	+24V IN	+24V DC supply input
16	0 V (+24V)	Reference point for +24 V _{DC} I/O

17	Analog output 0	Programmable analog output, default setting: "Filtered Actual speed"	±10V/5mA
18	Analog output 1	Programmable analog output, default setting: "Filtered Motor current"	
19	0V	Analog output reference point	-
20	+10V	Reference voltage +10V, reference point: terminal 19	+10V/10mA
21	-10V	Reference voltage -10V, reference point: terminal 19	-10V/10mA
22	Digital input 4	Programmable digital input, configured as: "Els_Inc Ratio"	+30V 3.2mA @ 15V 5mA @ 24V 6.4mA @ 30V
23	Digital input 5	Programmable digital input, configured as: "Els_Dec Ratio"	
24	Digital input 6	Programmable digital input, default setting: "External fault"	
25	Digital input 7	Programmable digital input, default setting: "Drive reset"	
26	Digital output 2	Programmable digital output, default setting: none	+30V/25mA
27	Digital output 3		
28	Digital output 4		
29	Digital output 5		
30	Relay-N0	"Drive OK" N.O. contact	250 V AC 1A AC
31	Relay-NC	"Drive OK" N.C. contact	
32	Relay-COM	"Drive OK" common contact	

7.3 Download Firmware

The standard firmware loaded at the factory is an application called PXBasic. The PXBasic application firmware is composed of 2 files:

- the low level firmware or firmware library (PXBasicVX_XX.sre).
- the parameters file, used by the user for the drive application settings (PXBasicVX_XX.par).

While firmware in the drive can be downloaded in the field by a customer when necessary, it is normally never required, and certainly not for routine use or configuration of the drive. To perform a firmware upgrade, refer to this AFTER contacting the factory.

- Open the configurator *E@syDrivesPX*.
- Enable communication with the drive from the menu “Target/connect”.
- From the menu “Service/Load firmware” select the command “Browse”.
- The file PXBasicVX_XX.sre of the last version will be indicated default. Choose it and select the command “Load”.
- At this time the firmware download is activated, and the data quantity (number Bytes) transferred is shown on the screen. The PXBasic is composed of nearly 170 kbytes and the download time is around 90 sec.
- Reset the drive with the command of reset from configurator, or turn off and on the 24 vdc supply.
- In the menu “File/Open” open the PXBasicVX_XX.par file.
- In the menu “Parameters/Write all” copy all the system parameters in the drive.
- In the menu “Parameters/Save parameters” save all the parameters in the drive.
- Reset the drive with the command “reset” configurator, or turn off and on.

At this time the firmware updating has finished, now the user can proceed setting the drive as desired.

7.4 Automatic Electric Phasing Procedure for Encoder/Resolver

The knowledge of the right phase relation between the current and the motor magnetic angle is fundamental for the drive performances. The simple electric and automatic phasing sequence of the FlexMax drive allows it to store the phasing angle in a drive parameter (electric phasing) in order to constantly supply precise information about the phase of the position/speed motor feedback (encoder/resolver).

Such procedure has to be performed every time the FlexMax drive is used with non-PowerTec motors. All PowerTec motors, on the contrary, are factory-phased (mechanical phasing). Before performing the automatic electric phasing, it is advisable to check the encoder/resolver connections (as described in the paragraph "Encoder Control/Drive Connections") and the power/U-V-W phase sequence connections.

Procedure

If this procedure is performed using the software of the *E@syDrivesPX* configurator, the following sequence has to be respected:

1. Start the software of the *E@syDrivesPX* configurator (from the Windows Start menu)
2. Enable the "MONITOR Window" function
3. Display in MONITOR Window the SYS_ENC_MECH_OFFSET parameter (IPA 20058) and the SYS_ENC_OFFSET parameter (IPA 20057) from the Service->Phasing menu

PowerTec

4. Remove any mechanical coupling from the motor shaft, so that it can move freely
5. Set the SYS_MOT_NOM_CURRENT parameter (IPA 20001) with the value referring to the motor rated current (from the 20-Motor data menu)
6. Set the SYS_APP_SEL parameter (IPA 18140) as "Phasing" (from the "Service" menu)
7. Save the parameters (Command "Save parameter into target").
8. Perform the command "Drive Reset" or switch the drive off and on again
9. Enable the drive using the Digital 0 Input
10. Check that the drive performs a current ramp till the limit set in the SYS_MOT_NOM_CURRENT parameter (IPA 20001) while the motor rotor carries out a small movement
11. After a few seconds the motor starts rotating and stops in a fixed position after performing a revolution. If the motor is PowerTec-marked, make sure that it rotates in a clockwise direction (from the motor shaft side); with non-PowerTec motors, check the wiring on the power cables between the drive and the motor.
The counting of the encoder/revolver must increase (see paragraph "Encoder Control/ Drive Connections") during the motor rotation (if PowerTec motor).
12. Check the value of the SYS_ENC_MECH_OFFSET parameter (IPA 20058) keeping the drive enabled. If the motor has been supplied by Powertec, it should be factory phased (mechanically) with the FlexMax drive, and the parameter value should be near the appropriate value from the following table (values in the range of ± 4 degrees are allowed):

4 pole motor	+90
6 pole motor	0
8 pole motor	-90

13. Perform the command "Save parameters into target" by keeping the drive enabled. The current value of the phasing angle is stored in the SYS_ENC_MECH_OFFSET parameter (IPA 20058)
14. Disable the drive
15. Set the APPLICATION SEL parameter (IPA 18140) (from the "Service" menu) with the original selection "Basic (Plc)" (factory default) or "Plc" (in case the FlexMax drive is used with a custom application developed in the MDPlc environment according to the IEC 61131-3 standard)
16. Save the parameters (command "Save parameters into target")
17. Use the command "Drive Reset" or switch the drive off and on again

At the end of this electric and automatic phasing procedure, it is suggested to configure the FlexMax drive with a speed mode and to check the motor functioning procedure.

7.5 Integrated CANopen Interface

CiA : CAN in Automation, user international group.

CAN : Controller Area Network.

CANopen is a communication profile for CAL-based industrial systems. The reference document is the CANopen CAL-Base COMMUNICATION PROFILE for Industrial Systems; CiA Draft Standard 301 Version 3.0. Issue October 1996 by CAN in Automation e. V.

The CAN protocol (ISO 11898) is CAN2.0A with an 11-bit identifier.

The integrated CANopen interface is developed as a “Minimum Capability Device”.

The data exchange is cyclic; the Master unit reads the Slave input data and writes the Slave output data; the Baud Rates for the interface are those foreseen by the CANopen specification.

7.5.1 CANopen Functions

This chapter describes the controlled functions of the CANopen communication profile. The main features are:

- 1) The “Minimum Boot-up” is managed; the “Extended Boot-up (CAL)” is not managed.
- 2) The SYNC function is implemented.
- 3) The PDO asynchronous assignment and RTR are managed.
- 4) The Node Guarding is managed.
- 5) The emergency message is managed (“EMERGENCY”).
- 6) The Dynamic ID distribution function (DBT slave) is not managed.
- 7) A “Pre-Defined Master/Slave connection” is implemented to simplify the Master tasks during the initialization phase. “Inhibit-Times” (in units of 100 uS) can be modified up to a value of 1 min.
- 8) The high-resolution synchronization is not supported.
- 9) “TIME STAMP” is not managed.
- 10) On the access of the structured parameters, the OFFhex option subindex (access to the whole object) is not managed.
- 11) In order to obtain a higher efficiency level, only the “Expedited” data transfer (max. 4 Bytes) of the SDO services is managed.

PDO: Process Data Object, service messages without confirmation used for the real time data transfer from/to the device.

DBT: Distributor. It is a service element of the CAN Application Layer in the CAN Reference Model; the DBT has the task to assign COB-ID to the COBs used by the CMS.

SDO: Service Data Object, service messages with confirmation used for the a-cyclic data transfer from/to the device.

7.5.1.1 Pre-defined Master/Slave Connection

The “Pre-defined Master/Slave connection” allows a peer-to-peer communication between one Master and 127 Slaves; the Broadcast address is zero.

NMT: Network Management. It is a service element of the CAN Application Layer in the CAN Reference Model; it initializes, configures and controls the errors of a CAN network.

7.5.1.2 NMT Services (Network Management)

The NMT “mandatory” services are:

- Enter_Pre-Operational_State CS = 128
- Reset_Node CS = 129
- Reset_Communication CS = 130

Being that the “Minimum Boot-up” is used, also the following NMT services are managed:

- Start_Remote_Mode CS = 1
- Stop_Remote_Mode CS = 2

The COB-ID * of an initialization NMT service is always at 0; CS is the Command Specifier defining the NMT service.

CS: Command Specifier; it defines the NMT service.

COB = Communication Object (CAN Message). It is a transport unit inside a CAN network. The data must be sent in network inside a COB.

COB-ID = COB-Identifier. It identifies a COB inside the network. It also states the COB priority.

7.5.1.3 Initialization

The FlexMax drive supports the Node Guarding mechanism. The Node Guarding configuration can be performed through the master via the standard Object Dictionary elements (1006h, 100Ch, 100Dh).

The drive checks the master functioning conditions through the Life Guarding. If the check fails, the drive enables the "Buss Loss" alarm. The Life Guarding threshold can be calculated as follows:

Value/Condition

60ms

Default. No parameterization of the Node Guarding.

SYNC_PERIOD (*)

LIFE_TIME_FACTOR

Use of the synchronous mode. If not stated by the master, the LIFE_TIME_FACTOR default value is equal to 3.

NODE_GUARDING_PERIOD (*)

LIFE_TIME_FACTOR

NODE_GUARDING_PERIOD set by the master

If not otherwise stated, the LIFE TIME FACTOR value is equal to 3

7.5.1.4 Communication Object

This chapter describes the communication objects of the CANopen protocol; they are managed by the interface card. The managed communication objects are:

- 1) 1 SDO reception Server.
- 2) 1 SDO transmission Server.
- 3) 2 reception PDOs.
- 4) 2 transmission PDOs.
- 5) 1 Emergency Object.
- 6) 1 Node Guarding - Life Guarding.
- 7) 1 SYNC object.

The following table lists the used communication objects with their priority level and the Message Identifier; the “Resulting COB-ID” is obtained by adding the Node-ID (card address) to the number.

Table 7.5.1: Communication Objects

OBJECT	PRIORITY	MESSAGE ID
1st SDO rx	6	1536
1st SDO tx	6	1408
1st PDO rx	2	512
1st PDO tx	2	384
2nd PDO rx	2	768
2nd PDO tx	2	640
EMERGENCY	1	220
NODE GUARDING	not used	1792
SYNC	0	128

Node Guarding has no priority because it is a special NMT service; it has the Message-ID because it is not a Broadcast service.

7.5.1.5 Object Dictionary Elements

The object dictionary can be accessed only via a CANopen master. Such elements, therefore, can not be modified via the configurator or the keypad. They can not even be permanently saved in the drive flash memory.

The following table lists the used communication objects:

Table 7.5.2: Objects used by the CANopen communication profile

Index (hex)	Name
1000	Device Type
1001	Error Register
1002	Manufacturer status register
1005	COB-ID SYNC Message
1006	Communication cycle period
1008	Manufacturer Device Name
1009	Manufacturer Hardware Version
100A	Manufacturer Software Version
100C	Guard Time
100D	Life Time Factor
100F	Number of PDOs supported
1014	COB-ID Emergency
1200	1st Receive SDO parameter

7.5.1.6 Rx PDO Entries

The structure of the PDO Communication Parameter (index 1400h, 1401h) is:

- 1) Subindex 0 (Number of supported entries) = 2
- 2) The structure of Subindex 1 (COB-ID used by the PDO) is:
 - Bit 31 (valid/invalid PDO) can be set via SDO.
 - Bit 30 (RTR Remote Transmission Request) = 0 because this function is not supported.
 - Bit 29 = 0 because the 11-bit ID is used (CAN 2.0A).
 - Bits 11-28 are not used.
 - Bit 0-10 COB-ID (see table 7.5.1).
- 3) Cyclic-synchronous Subindex 2 (Transmission Type), or synchronous according to the master performed setting (1 if SYNC has been foreseen, 254...255 if asynchronous). If not stated, the synchronous mode is active.

7.5.1.7 Tx PDO Entries

The structure of the PDO Communication Parameter (index 1800h, 1801h) is:

- 1) Subindex 0 (Number of supported entries) = 3
- 2) The structure of Subindex 1 (COB-ID used by the PDO) is:
 - Bit 31 (valid/invalid PDO) can be set via SDO.
 - Bit 30 (RTR Remote Transmission Request) = 0 because this function is not supported.
 - Bit 29 = 0 because the 11-bit ID is used (CAN 2.0A).
 - Bits 11-28 are not used.
 - Bit 0-10 COB-ID (see table 7.5.1).
- 3) Cyclic-synchronous Subindex 2 (Transmission Type), or synchronous according to the master performed setting (1 if SYNC has been foreseen, 254...255 if asynchronous). If not stated, the synchronous mode is active.
- 4) Inhibit time

7.5.1.8 SDO Entries

Only the “Expedited” data transfer mode (max. 4 Bytes) is used.

The structure of the SDO Communication Parameter is:

- 1) Subindex 0 (Number of supported entries) = 3 because the device is a Server of the SDO service.
- 2) The structure of the Subindex 1 and 2 (COB-ID used by the SDO) is:
 - Bit 31 (valid/invalid SDO); it is equal to 1 because just the Default SDOs are used.
 - Bit 30 reserved = 0.
 - Bit 29 = 0 because the 11-bit ID is used (CAN 2.0A).
 - Bits 11-28 are not used.
 - Bit 0-10 COB-ID (see table 7.5.1).

The element “node ID of SDO’s client resp. server” is not supported because just the Default SDOs are used.

7.5.1.9 COB-ID SYNC Entries

The structure of the 32 bits contained in the COB-ID SYNC communication parameter is:

- Bit 31 = 1 because the CANopen interface card is a “consumer” of SYNC messages.
- Bit 30 = 0 because the interface card does not create SYNC messages.
- Bit 29 = 0 because the 11-bit ID is used (CAN 2.0A).
- Bits 11-28 are not used.
- Bit 0-10 COB-ID (see table 7.5.1).

7.5.1.10 COB-ID Emergency

The structure of the 32 bits contained in the COB-ID Emergency Message communication parameter is:

- Bit 31 = 0 because the CANopen interface card is not a “consumer” of Emergency messages.
- Bit 30 = 0 because the interface card creates Emergency messages.
- Bit 29 = 0 because the 11-bit ID is used (CAN 2.0A).
- Bits 11-28 are not used.
- Bit 0-10 COB-ID (see table 7.5.1).

7.5.2 CANopen Management

The user interface of the CANopen protocol is performed via the drive parameters. The parameters are controlled via hierarchical menus. All the writing parameters referring to the field bus are active only after the drive reset. Here following is a list of drive parameters useful to control the CANopen protocol.

Fieldbus menu

The CANopen protocol can be enabled by setting the IPA 40000 SYS_FB_TYPE parameter as "Can Open". The other parameters of this menu are:

IPA	Par. Name	Type	Default value	Attr.
40100	SYS_FB_ADDRESS	1 byte unsigned	0	Write
40001	SYS_FB_BAUD_RATE	4 bytes unsigned	0	Write
40110	SYS_FB_CC_ENABLED	Enum	Enabled	Write
40111	SYS_FB_PDC_ENABLED	Enum	Enabled	Write
40114	SYS_FB_FAIL_CAUSE	4 bytes unsigned	0	Read only

- “SYS_FB ADDRESS” = address of the node;
- “SYS_FB BAUD_RATE” = network baud rate. The baudrate is stated directly in baud (ex. 125kb = 125000);

- “SYS_FB_PDC_ENABLED” e “SYS_FB_CC_ENABLED” = allow the user to enable/disable the corresponding channels. With the PDC channel it is possible to exchange up to 8 parameters

The “SYS_FB_FAIL_CAUSE” parameter defines the error cause. Presently the following causes are provided:

Cod.	Meaning
1	Protocol type not correct
2..8	Protocol not available
18 .. 24	configuration error on the reception PDC channel
25..31	configuration error on the transmission PDC channel
32	too many rx bytes on the PDC channel
33	too many tx bytes on the PDC channel
100	Baud rate not correct
101	Node address not correct
103	Non expedited SDO type not supported
104	SDO length not correct
105	Error on NMT messages
106	NMT code non-existent
107	Can line on “Bus-off” status
108	NMT modality not supported

7.5.3 Process Data Channel Control

This function allows to allocate the drive parameters or application variables to the Process Data Channel data. As for the CANopen protocol, the PDC is performed via the PDO messages (Process data Object). The CANopen protocol uses a number of words for the Process Data Channel (abbr. PDC Process Data Channel), which can always be set. The fieldbus Process Data Channel configuration is the following:

Data 0 Data... Data n

The drive can both read and write the Process Data Channel data. A datum can be made both of 2 and 4 bytes. The word "data" refers to any quantity of bytes included between 0 and 8, if the byte total number required is not higher than 16.

Example

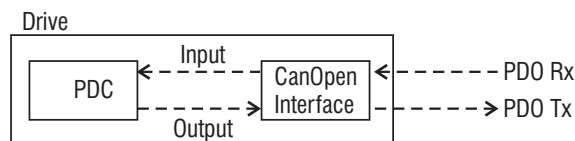
It is possible to have:

- from 0 to 8 data with 2 bytes
- 1 datum with 4 bytes + from 0 to 6 data with 2 bytes
- 2 data with 4 bytes + from 0 to 4 data with 2 bytes
- 3 data with 4 bytes + from 0 to 2 data with 2 bytes
- 4 data with 4 bytes

The data exchanged via the PDC can be of two types: drive parameters and variables of an MDPlc application. The use of the MDPlc variables is described in par. 7.5.3.3

The composition of the PDC input and output data is defined via suitable parameters as described in the paragraphs 7.5.3.1 and 7.5.3.2.

The master writes the data defined as PDC input and reads the data defined as PDC output.



7.5.3.1 PDC Input Configuration (FB XXX MS Parameter)

The configuration of the PDC input channel can be performed via 8 menus with the same structure.

IPA	Par. Name	Typo	Default value	Attr.	Unit
40190	SYS_FB_ASSIGN_M2S_1	Enum	Not assign	Writing	-
40200	SYS_FB_IPA_M2S_1	2 bytes unsigned	0	Writing	-
40210	SYS_FB_FORMAT_M2S_1	Enum	16 bit int	Writing	-
40220	SYS_FB_EXP_M2S_1	2 bytes unsigned	16 bit integer	Writing	-

This structure refers to the first input parameter. The structure is repeated 8 times for the 8 possible input parameters. The following parameter indexes are 40201..40221, 40202..40222 etc.

The "SYS_FB_ASSIGN_M2S_1" parameter can be selected as follows:

- **Parameter:** the PDC corresponding datum is combined to a parameter identified by "SYS_FB_ASSIGN_M2S_1". The parameters are entered into engineering units and are exchanged in an asynchronous way.
The "SYS_FB_FORMAT_M2S_1" parameter sets the parameter writing format. The format can be different from the parameter original one.
The "SYS_FB_EXP_M2S_1" parameter defines the 10th power which the parameter is multiplied by before being transferred to the drive.

A practical example for the parameter use:

The 32001 ELS_RATIO_0 parameter, with a float format is written by the master. It must be written with an integer format, signed and three decimal digits. Set the parameters as follows:

```
40200 SYS_FB_IPA_M2S_1 @ 32001
40210 SYS_FB_FORMAT_M2S_1 @ "16 bit integer"
40220 SYS_FB_EXP_M2S_1 @ 3
```

In this way the master must write:

```
1000 to set the value 1.000
-1234 to set the value -1.234.
```

- **Direct Access:** the PDC corresponding datum is combined to a parameter identified by "SYS_FB_ASSIGN_M2S_1".
The parameters are entered into internal counts and are exchanged in an asynchronous way (one every 8 msec). The writing format identified by the "SYS_FB:FORMAT_MS2_1" parameter (see the table in the 230-Fieldbus menu) must coincide with the drive internal format.
As for parameters with a float internal format, it is possible to choose "32 bit integer" and the conversion into a float format between the received datum and the internal datum is performed automatically.

Before establishing the Profibus communication between the Master and the drive, it is necessary to assign the drive parameters to the Process Channel. These parameters can be activated by resetting the drive.

7.5.3.2 PDC Output Configuration (FB XXX SM Parameter)

The output configuration of the PDC channel can be performed via 8 menus with the same structure.

IPA	Par. Name	Type	Default value	Attr.	Unit
40290	SYS_FB_ASSIGN_S2M_1	Enum	Not assign	Writing	-
40300	SYS_FB_IPA_S2M_1	2 bytes unsigned	0	Writing	-
40310	SYS_FB_FORMAT_S2M_1	Enum	16 bit int	Writing	-
40320	SYS_FB_EXP_S2M_1	2 bytes unsigned	16 bit integer	Writing	-

This is the structure for the first output parameter. The structure is repeated 8 times for the 8 possible output parameters. The indexes of the following parameters are 40301..40321, 40302..40322 etc.

The "SYS_FB_ASSIGN_S2M_1", "SYS_FB_IPA_S2M_1", "SYS_FB_FORMAT_S2M_1" and "SYS_FB_EXP_S2M_1" parameters have the same meaning as those described in point 7.5.3.1.

7.5.3.3 Use of the PDC in MDPlc Applications

It is possible to configure both the PDC input and output data in order to allow the data direct access via the MDPlc application code.

For more details see the manual "Drive programming with MDPlc" on "FlexMax tools" cd-rom.

7.5.4 SDO Management

The SDO service is available only if the 40110 "SYS_FB_CC_ENABLED" parameter is ON.

The drive parameters can be accessed via the "MSPA" Manufacturer Specific Profile Area (2000hex< index <5FFFhex).

As the drive parameter indexes (IPA) normally exceed the CANopen MSPA, the FlexMax drive is supplied with an offset value allowing the access to the drive parameters. The index to be stated in the SDO command to access a drive parameter can be obtained via the following formula:

$$\text{SDO index} = 2000 \text{ hex} + \text{IPA-OFFSET}$$

The OFFSET value can also be accessed (and modified) via the 5FFF hex index of the CANopen Object Dictionary. The default value is 20000.

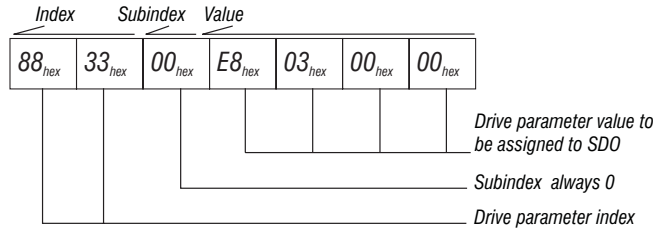
The Subindex field has always to be set with 0, if not, an error message is displayed. The Data field must contain the value of the drive parameter.

Example:

Writing the value 1000 in the 23000 (SYS_VEL_FAK) parameter.

The required information is:

- 1) The SDO index resulting from the formula is
 $2000 \text{ hex} + 23000 \text{ dec} - 18000 \text{ dec} = 13192 \text{ dec} (3388 \text{ hex})$
- 2) The value to be written is 1000, corresponding to 03E8 hex.



In case an error occurs during the parameter reading or setting, the CANopen interface sends an Abort domain transfer message; the value of Application-error-codes has the following meanings:

Error class	Error code	Additional code (hex)	Meaning
6	0	0	Parameter doesn't exist
8	0	22	Access failed because of present device state
6	1	2	Read/Write only error
8	0	0	Generic error
6	9	32	Minimum value
6	9	31	Maximum value
5	4	0	SDO time_out
5	4	1	Invalid command
3	9	30	Invalid value

7.5.5 Alarms

Fieldbus alarms

The bus failure is signaled via the 26- "Field bus failure" alarm. As for CANopen, the possible failure causes are:

- "Bus-off" condition of the CAN line;
- th drive has not been enabled in the "Operational" mode;
- the "Life Guarding" threshold has been overcome.

This alarm becomes active only when the drive is enabled.

If ON, the 40115 "SYS_FB_ALARM_CARE" parameter enables the generation of the "Field bus failure" alarm also when the drive is disabled.

Drive alarm handling

Considering that the fieldbus must function with different firmware application systems, the "drive alarm status" is not foreseen.

The "drive alarm status" is not therefore given any special treatment.

The FlexMax firmware, version 2.1 and later, provides a series of parameters capable of detecting the drive status.

Alarm reset

The alarm reset is one of the drive standard functions, i.e. each application provides the same parameter for this function. It is therefore possible to reset the alarms via the configuration channel on the firmware of all the different drives. The alarm reset can be performed by sending the value 1 to the parameter 18012.

The FlexMax firmware, version 2.1 and later, provides the "Virtual Digital Input" function, through which it is possible effect a bit-controlled alarm reset.

7.6 Modbus

7.6.1 Modbus Protocol and Message Format

Refer to “MODBUS RTU Protocol, chapters 1 and 2” Instruction manual.

NOTE! Do not use address 0 in the Modbus protocol (SYS_SL3_ADDR, IPA 18031) since it is reserved for broadcast command.
Set "SYS_SERPROT" (IPA 18032) as "Modbus".

7.6.2 Modbus Functions

The following functions are implemented on the drive:

Code	Function	Description
01 (*)	Read coil status	This function allows to require the ON or OFF condition of the Drive discrete parameters (Coil). The broadcast mode is not allowed.
02 (*)	Read input status	This function allows to require the ON or OFF condition of the Drive discrete parameters (input). The broadcast mode is not allowed.
03 (*)	Read holding registers	This function allows to require the value of 16-bit (word) registers containing Drive parameters. The broadcast mode is not allowed.
06	Preset single register	This function allows to set the value of a single 16-bit register. The broadcast mode is allowed.
16 (*)	Preset multiple registers	This function allows to set the value of a consecutive block made of 16-bit registers. The broadcast mode is allowed.

Note: For a detailed function description refer to “MODBUS RTU Protocol, chapter 3” Instruction manual
(*) Multiple request cannot be executed. Only one parameter can be accessed at the time.

NOTE! The 16-bit Drive parameter (word or integer type) is referred to as 16-bit Modbus register; a 32-bit Drive parameter (Dword, long or float type) covers therefore two Modbus registers. For the float format, the first word is the most significative part of the 32-bit data. For the Dword or long format, the first word is the less significative part of the 32-bit data. Each word is the register. The registers require two bytes where the first one contains the most significative section.

7.6.3 Error Management

Refer to “MODBUS RTU Protocol, chapter 4” Instruction manual.

7.6.3.1 Exception codes

The protocol implemented on the drive foresees the following exception codes.

Code	Name	Meaning
00	ILLEGAL ADDRESS	Address is not valid.
01	ILLEGAL FUNCTION	The received function code does not correspond to a function allowed on the addressed slave.
02	ILLEGAL DATA ADDRESS	The address number, which the data field refers to, is not a register allowed on the addressed slave.
03	ILLEGAL DATA VALUE	The value to be allocated, which the data field refers to, is not allowed for this register.
04	SLAVE FAIL	The Slave cannot execute the requested command
05	SLAVE ACK	The Slave has accept and is executing the requested command
06	SLAVE BUSY	The Slave is busy
07	NAK - NEGATIVE ACKNOWLEDGEMENT	The function can not be performed with the present operating conditions or attempt to write an only-reading parameter.

IMPORTANT! The settings of SYS_BAUDRATE (IPA 20024) is enabled with the drive start-up; it is therefore required to store it and to switch the drive off.

7.6.4 System Configuration

No configuration is necessary in order to use Modbus Protocol on the drive. The switching between SLink 3 and Modbus Protocol is automatically set on the drive.

The serial port configuration is managed by the 015-Comm Config menu (parameters: SYS_SL3_ADDR (IPA 18031), SYS_BAUDRATE (IPA 20024), SYS_SERCFG (IPA 20025), SYS_DRIVE SER DELAY TIME (IPA 20026) and SYS_SERPROT“ (IPA 18032).

In order to communicate with the drive through the E@syDrives configurator in Modbus Protocol, it is necessary to set “Modbus” in the “Communication setup” on Target windows.

7.6.5 Appendix - Register and Coil Modbus Tables

In the drive the **register number** and **parameter index** (IPA) are the same.]

7.7 DeviceNet Interface (XVy-DN)

This chapter describes the software for connecting of FlexMax drives to DeviceNet networks.

It is intended for design engineeres and technicians responsible for the maintenance, commissioning and operation of DeviceNet systems.

A basic knowledge of DeviceNet is assumed and may be found in the following manuals:

- DeviceNet Specifications. Volume 1 - DeviceNet Communication Model and Protocol (Issued by ODVA).
- DeviceNet Specifications. Volume 2 - DeviceNet Device Profiles and Object Library (Issued by ODVA)

7.7.1 DeviceNet General Description

DeviceNet is a profile of communication for industrial systems based on CAN.

As protocol CAN (ISO 11898) is used CAN2.0A with the 11 bit identifier.

TheXVy-DN driver is developed as “Slave UCMM Capable Device” for operating only in “Predefined Master/Slave Connection Set”.

The data transfer is carried out cyclically; the Master unit reads the data supplied by the Slaves and writes the Slave reference data; the Baud Rates supported are:

- 125 kbit
- 250 kbit
- 500 kbit .

The physical support is given by the RS485 serial line; a maximum of 64 Slaves can be connected to the Bus.

7.7.2 Connection

The CAN terminals allows to connect the FlexMax drive to DeviceNet network. Refer to chapter 4.3.6 of this manual for more details.

7.7.3 Leds

The DeviceNet connection leds are behind the CAN connector.

Name	Color	Function
CAN	Green	The led is ON when the connection is powered (pin C1, C5)
AL	Red	DeviceNet connection status see next table
OP	Green	DeviceNet connection status see next table

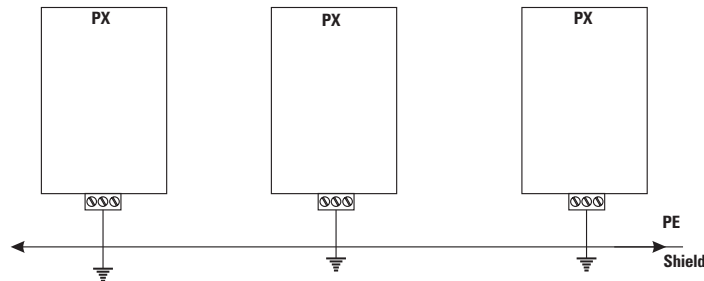
Table 7.7.3.1: AL-OP leds status codification

OP	AL	Meaning
ON	ON	Card power-up
BLINK	BLINK	Self test and Duplicate MAC-ID check is running
BLINK	OFF	Master configuration and/or I/O Polling wait not active
ON	OFF	I/O Polling active, operative status
OFF	BLINK	Minor fault (DUP MAC-ID fail, bus-off, bus-loss)
OFF	ON	Major fault (configuration error, internal error)
OFF	OFF	DeviceNet not configurated

7.7.4 Interface

For the connection to the Bus please use a shielded twisted cable recommended by DeviceNet specification.

The connection among the single drives is accomplished by a shielded cable as shown in the following figure:



7.7.5 DeviceNet Function

In this chapter are described the functions of DeviceNet managed by the driver. The main characteristics are:

1. XVy-DN operates only as Slave in “Predefined Master/Slave Connection Set”.
2. Within the “Predefined Master/Slave Connection Set” the driver is a “UCMM Capable Device”.
3. The “Explicit Messaging” is managed.
4. The “Polling” for the fast cyclical data exchange Master/Slave is managed.
5. The detection mechanism of the “Duplicate MAC ID” is implemented.

Regarding the “Explicit Messaging” the fragmentation of the data frame, with a total of max. 32 byte, is managed.

Connection sizes

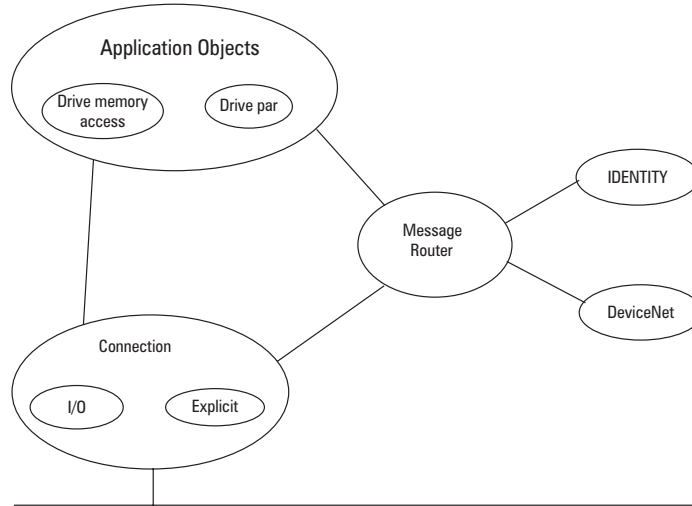
CONNECTION INSTANCE	PRODUCED	CONSUMED
Polled I/O	Depending on frame setting	
Explicit messaging	32	32

7.7.6 Object description

Hereafter you find the description of the objects managed by the XVy-DN driver.

7.7.6.1 Object Model

The following figure shows the XVy-DN “Object Model”.



The following table shows:

1. The object classes of XVy-DN driver.
2. If the class is mandatory.
3. The number of instances included in every class.

See “DeviceNet Specifications” for the Standard classes.

Object	Optional/Required	# of Instances
Identity	Required	1
Message Router	Required	1
DeviceNet	Required	1
Connection	Required	1 I/O, 3 Explicit
Drive Parameter Access	Optional	many
Drive memory Access	Optional	many

7.7.6.2 How Objects Affect Behavior

The “Affect Behaviour” of the objects is reported in the following table:

Object	Effect on Behavior
Identity	Supports “Reset Service”
Message Router	No effect
DeviceNet	Port attributes configuration
Connection	Contains the number of logical ports
Drive Parameter Access	Drive parameters read/write
Drive Memory Access	Drive parameters read/write

7.7.6.3 Defining Object Interface

The object interface of the XVy-DN driver is the following:

Object	Interface
Identify	Message router
Message Router	Explicit Messaging Connection Instance
DeviceNet	Message router
Connection	Message router
Drive Parameter Access	Message router
Drive memory Access	Message router

7.7.7 Data transfert via Explicit Messaging

The data transfer via Explicit Messaging is made through two new objects: one for accessing the Drive parameters, the other to direct access the drive memory.

7.7.7.1 Drive Parameter Access

For reading/writing the drive parameters the Drive Parameter Access object is defined with the following characteristics:

- Class ID: 66h.
- Class Attribute: Revision
- Instance Attribute: This instance does not provide any attribute.

7.7.7.1.1 Class Code

Class code: 66 hex

7.7.7.1.2 Class attributes

Number	Need in implementation	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Semantics of values
1	Optional	Get	Revision	UINT	Revision of this object	

dn345

7.7.7.1.3 Instance Attributes

Number	Need in implementation	Access Rule	Name	DeviceNet Data Type	Description of Attribute	Semantics of values
This instance does not provide attributes						

dn350

7.7.7.1.4 Common Services

This object has no common services.

7.7.7.1.5 Object Specific Services

Service Code	Need in implementation		Service Name	Description of Service
	Class	Instance		
32 _{hex}	n/a	Required	Get_Drive_Value	Read drive parameter value
33 _{hex}	n/a	Required	Set_Drive_Value	Writes drive parameter value
34 _{hex}	n/a	Required	Get_Typed_Drive_Value	Read drive parameter value indicating the data type
35 _{hex}	n/a	Required	Set_Typed_Drive_Value	Writes drive parameter value indicating the data type

dn355

7.7.7.1.6 Behavior

This object is the interface between the DeviceNet network and all Drive parameters. The access to the Drive parameter is carried out by the parameter index; if the parameter does not exist or may not be accessed for any reason (for example: try to write a read only parameter) an error code will be returned. Drive parameters in text format cannot be accessed. In the following are repeated patterns of how the data frame of data has to be composed for reading/writing Drive parameters.

A) Write Drive Parameter

In this example the writing of a Drive parameter is shown; the cases of positive or wrong writing are distinguished.

A-1) Write Drive Parameter Request

The data frame for writing a drive parameter is composed as follows:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	33hex	Set Drive Parameter - Object Specific Service
See Note ¹⁾	Class ID	66hex	Drive Parameter Access Class Object
	Instance ID	XXXX	Drive Parameter Index in format Low byte-High byte
Byte ²⁾	VALUE	XX	Low byte-Low word drive parameter value
		XX	High byte-Low word drive parameter value
		XX	Low byte-High word drive parameter value
		XX	High byte-High word drive parameter value

dn360

- 1) Byte or Word depending on the type of allocation executed by the Master.
- 2) The number of bytes of the "Value"-field depends on the length of the Drive parameter; i.e.: if the Drive parameter type is "Integer" the length of VALUE is 2 bytes.

A-2) Write drive parameter - Reply OK

If the Drive parameter is written correctly, the response is:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	33hex OR 80hex	Set Drive Parameter Reply code- Object Specific Service.
Word	Result	0000	Result field equal to zero means writing correctly executed.

dn365

A-3) Write drive parameter - Reply Error

If the writing of the drive parameter has been rejected, the response is the following:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	33hex OR 80hex	Set Drive Parameter Reply code- Object Specific Service.
Word	Result	XXXX ¹⁾	Drive specific error code.

dn370

- 1) For error codes see table 7.7.7.1. .

B) Read Drive Parameter

In this example is shown the reading of a Drive parameter; the cases of positive or wrong reading are distinguished.

B-1) Read Drive Parameter Request

The data frame for the Drive parameter reading is composed as follows:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	32hex	Get Drive Parameter - Object Specific Service.
See Note ¹⁾	Class ID	66hex	Drive Parameter Access Class Object.
See Note ¹⁾	Instance ID	XXXX	Drive Parameter Index in format Lowbyte-High byte.

dn375

1) Byte or Word depending on the type of allocation executed by the Master.

B-2) Read drive parameter - Reply OK

If the Drive parameter is read correctly, the response is:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	32hex	Get Drive Parameter Reply code- Object Specific Service.
Word	Result	0	Result field equal to zero means reading correctly executed.
Byte ¹⁾	VALUE	XX	Low byte-Low word drive parameter value.
			High byte-Low word drive parameter value.
			Low byte-High word drive parameter value.
			High byte-High word drive parameter value.

dn380

1) The number of bytes of the Value-field depends on the length of the Drive parameter; i.e. if the Drive parameter type is "Integer" the length of VALUE is 2 bytes.

B-3) Read drive parameter - Reply Error

If Drive parameter reading is rejected, the response is the following:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	32hex	Get Drive Parameter Reply code- Object Specific Service.
Word	Result	XXXX ¹⁾	Drive specific error code.

dn385

1) For error codes see table 7.7.7.1. .

C) Write Typed Drive Parameter

In this example the writing of a Drive parameter is shown; the cases of positive or wrong writing are distinguished.

In this case, it is shown the parameter IPA number, the value and the data type used in the data transmission.

The optional data type conversion is automatically executed by the firmware.

C-1) Write Drive Parameter Request

The data frame for writing a drive parameter is composed as follows:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	35hex	Set Drive Parameter - Object Specific Service
See Note ¹⁾	Class ID	66hex	Drive Parameter Access Class Object
	Instance ID	XXXX	Drive Parameter Index in format Low byte-High byte
Byte ²⁾	DATA TYPE	XX	Value data type
Byte ³⁾	VALUE	XX	Low byte-Low word drive parameter value
		XX	High byte-Low word drive parameter value
		XX	Low byte-High word drive parameter value
		XX	High byte-High word drive parameter value

1) Byte or Word depending on the type of allocation executed by the Master. ^{dn390}

2) The coding of the possible data type is listed in table 7.7.7.2.

3) The number of bytes of the "Value"-field depends on the length of the Drive parameter; i.e.: if the Drive parameter type is "Integer" the length of VALUE is 2 bytes.

C-2) Write drive parameter - Reply OK

If the Drive parameter is written correctly, the response is:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	33hex	Set Drive Parameter Reply code- Object Specific Service.
Word	Result	0000	Result field equal to zero means writing correctly executed.

dn395

C-3) Write drive parameter - Reply Error

If the writing of the drive parameter has been rejected, the response is the following:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	33hex	Set Drive Parameter Reply code- Object Specific Service.
Word	Result	XXXX ¹	Drive specific error code.

dn400

1) For error codes see table 7.7.7.1. .

D) Read Drive Parameter

In this example is shown the reading of a Drive parameter; the cases of positive or wrong reading are distinguished.

In this case, it is shown the parameter IPA number, the value and the data type used in the data transmission.

The optional data type conversion is automatically executed by the firmware.

D-1) Read Drive Parameter Request

The data frame for the Drive parameter reading is composed as follows:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	36hex	Get Drive Parameter - Object Specific Service.
See Note ¹⁾	Class ID	66hex	Drive Parameter Access Class Object.
See Note ¹⁾	Instance ID	XXXX	Drive Parameter Index in format Lowbyte-High byte.
Byte ²⁾	DATA TYPE	XX	Value data type

dn405

For parameter format see table 7.7.7.2.

- 1) Byte or Word depending on the type of allocation executed by the Master.
- 2) The coding of the possible data type is listed in table 7.7.7.2.

D-2) Read drive parameter - Reply OK

If the Drive parameter is read correctly, the response is:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	32hex	Get Drive Parameter Reply code- Object Specific Service.
Word	Result	0	Result field equal to zero means reading correctly executed.
Byte ¹⁾	VALUE	XX	Low byte-Low word drive parameter value.
			High byte-Low word drive parameter value.
			Low byte-High word drive parameter value.
			High byte-High word drive parameter value.

dn380

1) The number of bytes of the Value-field depends on the length of the Drive parameter; i.e. if the Drive parameter type is "Integer" the length of VALUE is 2 bytes.

D-3) Read drive parameter - Reply Error

If Drive parameter reading is rejected, the response is the following:

DATA TYPE	FIELD	VALUE	MEANING
Byte	Service Code	32hex	Get Drive Parameter Reply code- Object Specific Service.
Word	Result	XXXX ¹	Drive specific error code.

dn385

1) For error codes see table 7.7.7.1. .

Table 7.7.7.1: Error codes for the parameter access

RESULT	VALUE	MEANING
DB_E_OK	0	No error
DB_E_NO_IPA	-1	Parameter not exist
DB_E_SYSERR	-2	Generic error
DB_E_TYPE	-3	Type not supported
DB_E_READONLY	-4	Attempt to write a read only parameter
DB_E_NOTWRITENOW	-5	Attempt to write a parameter when not allowed
DB_E_MINVAL	-6	Value exceed minimum value
DB_E_MAXVAL	-7	Value exceed maximum value
DB_E_CNFCONFLICT	-8	Attempt to assign a currently invalid value
DB_E_CONSTANTLIMITS	-9	Attempt to access a parameter using currently invalid type

Table 7.7.7.2: Parameter format

FORMAT	VALUE	MEANING
DB_T_VOID	0	Return the value in the original format
DB_T_INT	1	16 bit signed
DB_T_WORD	2	16 bit unsigned
DB_T_LONG	3	32 bit signed
DB_T_DWORD	4	32 bit unsigned
DB_T_FLOAT	6	Float in IEEE 744 format

7.7.8 Polling Function

This type of DeviceNet-function is used for a fast cyclic exchange of Drive-parameters between Master and FlexMax drive.

The characteristics of the Polling-function are:

1. The data frame length is configurable through specific drive parameter (see chapter 10.3, 230-Fieldbus) and can vary from 1 to 7 word for both directions (Slave->Master and Master->Slave).
2. The card, as it is a Slave, during the Polling consumes Output data and produces Input data as response.

The configuration of the Drive parameters transferred via Polling is set by using configuration parameter allocated in the drive (see chapter 10.3, 230-Fieldbus).

7.7.9 XVy-DN Interface configuration

The DeviceNet interface configuration is performed via the drive parameters. The parameters are controlled via hierarchical menus. All the writing parameters referring to the DeviceNet interface are active only after the drive reset. Here following is a list of drive parameters useful to control the DeviceNet interface

7.7.9.1 Fieldbus Menu

The XVy-DN communication card can be enabled by setting the IPA 40000 SYS_FB_TYPE parameter as "Devicenet". The other parameters of this menu are:

IPA	Par. Name	Type	Default value	Attr.
40100	SYS_FB_ADDRESS	1 byte unsigned	0	writing
40001	SYS_FB_BAUD_RATE	4 bytes unsigned	0	writing
40110	SYS_FB_CC_ENABLED	Enum	Enabled	writing
40111	SYS_FB_PDC_ENABLED	Enum	Enabled	writing
40114	SYS_FB_FAIL_CAUSE	4 bytes unsigned	0	read only

- SYS_FB_ADDRESS (IPA 40100) = address of the node; admitted values 1 ... 63.
- SYS_FB_BAUD_RATE (IPA 40001) = network baud rate. The baudrate is stated directly in kbaud (ex. 125kb = 125); admitted values 125, 250, 500.
- SYS_FB_PDC_ENABLED (IPA 40111) and SYS_FB_CC_ENABLED (IPA 40110) = allow the user to enable/disable the corresponding channels. With the PDC channel it is possible to exchange up to 8 parameters.
- SYS_FB_FAIL_CAUSE (IPA 40114) = error cause. See the following table

7.7.9.2 Error Codes

Cod.	Meaning
2..8	hw or fw serious error. If the problem persists, replace the card
11..15	hw or fw serious error. If the problem persists, replace the card
18 .. 24	configuration error on the reception PDC channel
25..31	configuration error on the transmission PDC channel
32	too many rx bytes on the PDC channel
33	too many tx bytes on the PDC channel
34	the HS channel is disabled but rx contains some enabled parameters
35	the PDC channel is disabled but tx contains some enabled parameters
36	it is not possible to rx more than 4 parameters on the PDC channel
37-38	configuration error on the reception PDC channel
39	it is not possible to tx more than 4 parameters on the PDC channel
40-41	configuration error on the transmission PDC channel
100	baud-rate value not correct
101	Node address not correct
107	Bus-off
108	Internal error
109	Failed Duplicate MAC-ID check
110	DeviceNet interface not enabled

7.7.10 Alarms

7.7.10.1 XVy-DN Alarms

The XVy-DN interface provides two possible alarms:

Alarm 26 "Field Bus failure", is automatically enabled if there is no communication on the bus at a PDC level (polling I/O). This alarm becomes active only when the drive is enabled. If ON, the SYS_FB_ALARM_CARE parameter (IPA 40115) enables the generation of the "Field bus failure" alarm also when the drive is disabled.

7.7.10.2 Drive alarm handling

Considering that the card must function on different firmware application systems, the "drive alarm status" is not foreseen.

The "drive alarm status" is not therefore given any special treatment. The FlexMax firmware, version 2.1 and later, provides a series of parameters capable of detecting the drive status. See the drive manual for further information.

7.7.10.3 Alarm reset

The alarm reset is one of the drive standard functions, i.e. each application provides the same parameter for this function. It is therefore possible to reset the alarms via the configuration channel on the firmware of all the different drives. The alarms can be reset by sending the value 1 to the 18012 parameter.

The reset of the bit-controlled alarms can be performed also via the "Virtual Digital Input" function.

7.7.11 Process Data Channel Control

This function allows to allocate the drive parameters or application variables to the Process Data Channel data. The XVy-DN interface uses a number of words for the Process Data Channel (abbr. PDC Process Data Channel), which can always be set. The Process Data Channel configuration for the XVy-DN interface is the following:

DATUM 0 DATUM... DATUMn

The Slave can both read and write the Process Data Channel data. The DeviceNet data read by the Slave are defined as input data; the data written in DeviceNet by the Slave are defined as output data. A datum can be made both of 2 and 4 bytes. The word "data" refers to any quantity of bytes included between 0 and 8, if the byte total number required is not higher than 16.

Example

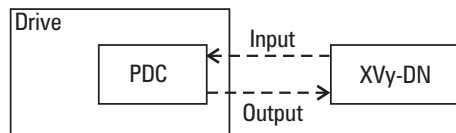
It is possible to have:

- from 0 to 8 data with 2 bytes
- 1 datum with 4 bytes + from 0 to 6 data with 2 bytes
- 2 data with 4 bytes + from 0 to 4 data with 2 bytes
- 3 data with 4 bytes + from 0 to 2 data with 2 bytes
- 4 data with 4 bytes

The data exchanged via the PDC can be of two types:

- drive parameters
- variables of an MDPlc application

The composition of the PDC input and output data is defined via suitable parameters as described in the paragraphs 7.7.11.1 and 7.7.11.2. The master cyclically writes the data defined as PDC input and cyclically reads the data defined as PDC output.



7.7.11.1 PDC Input Configuration (SYS_FB_XXX_MS parameter)

See paragraph 7.5.3.1.

7.7.11.2 PDC Output Configuration (SYS_FB_XXX_SM Parameter)

See paragraph 7.5.3.2.

7.7.11.3 Configuration of the Virtual Digital I/Os

The FlexMax firmware, version 2.1 and later, provides the "Virtual Digital I/O" function, which allows to exchange discrete signals between the master and the slave and vice versa. See the chapter 10.3 for a detailed description of these parameters. Other application firmware, for example MDPlc, does not provide the "Virtual Digital I/O" function.

7.7.11.4 Use of the PDC in MDPlc Applications

It is possible to configure both the PDC input and output data in order to allow the data direct access via the MDPlc application code. For more details see the manual “Drive programming with MDPlc” on “FlexMax tools” cd-rom.

NOTES: