NANO-A USER'S MANUAL

JETTER GmbH
Gräterstr. 2
D-71642 Ludwigsburg

Tel.	+49 7141 2550 0
Fax	+49 7141 2550 425
Hotline	+49 7141 2550 444
E-Mail	jetter@jetter.de
Mailbox	+49 7141 59834

Edition 1.0
September 1997

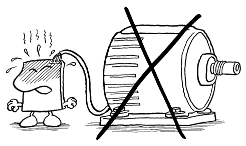
Table of Contents

I. Safety Instructions, General Technical Specifications

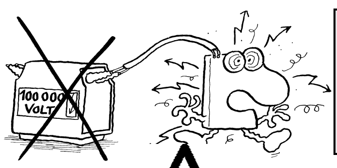
1. Safety Instructions

- The PROCESS-PLC NANO-A is a quality product, made according to the recognised electrotechnical rules.
  The device has been delivered by the manufacturing company in faultless state. In order to keep up this condition and to guarantee problem free operation, the technical specifications given in this documentation are to be observed.

- The devices must not be used for purposes other than the purposes they have been designed for.



- The devices are only to be used inside the limits given in their technical data.



- The devices are only to be operated by SELV. The maximum operating voltage must not be exceeded.

When failure or malfunctioning of the device could result in endangering of man or damage of equipment, this should be prevented by incorporating additional safety mechanisms, like limit switches, protection devices, etc., into the system.

Note:

The data indicated in this manual have got merely informational character without warranty of any quality.

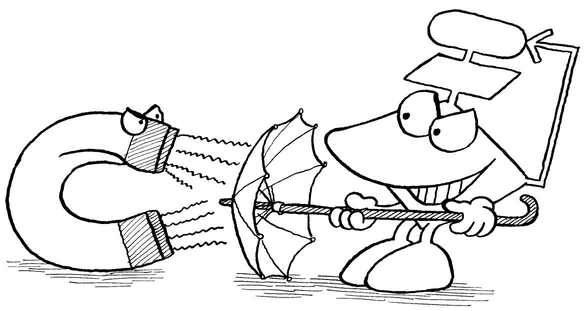
2. General Technical Specifications

Note:

The general technical specifications listed below apply to all PROCESS-PLC NANO-A modules. Above that, further module specific data will be mentioned in the respective chapters on modules.

Technical Data		Remarks
Ambient temperature	0 .. 50 °C	
Storing temperature	-10 .. 70 °C	
Air humidity	5% - 95%	RH-2 according to IEC 1131-2
Contamination level	II	according to IEC1131-2
Oscillation fatigue limit	IEC 1131-2	
Protective system	IP20	
Category of protection	III	according to IEC1131-2
ESD	Level ESD-4	according to IEC 1131-2
Housing	Aluminium	

3. Instructions on Electro-Magnetical Interference (EMI)



- A characteristic of interference immunity is the same as that of the often quoted chain: **It is as strong as its weakest link.**

Besides other precautions, shielding is important

- That's why - besides precautions inside the device - cable connections, respectively correct shielding, are of greatest importance.
- Shielding must be done on both ends of the applicable cables.
- The entire shield must be drawn behind the isolation, and then be extensively clamped under a strain relief.

Direct and extensive grounding is important

- When the signal is connected to terminal screws: The strain relief must be connected with a grounded surface directly and extensively.

Please use metallised male connector housings

- When male connectors are used: Only use metallised connectors, e.g. SUB-D with metallised housing. Please take care of direct connection here as well.

Separate signal and voltage connections spatially

- On principle, separate signal and voltage connections spatially.

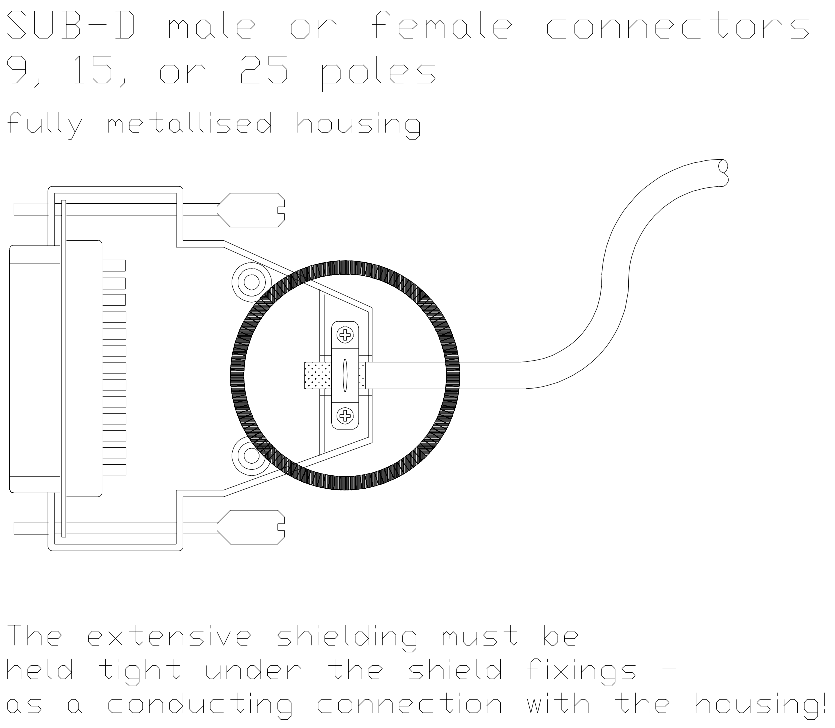
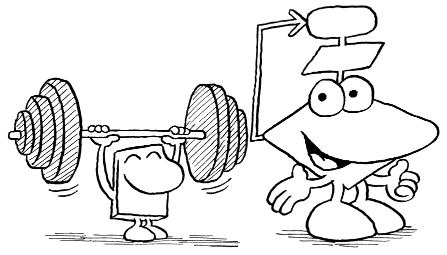

SUB-D male or female connectors
9, 15, or 25 poles
fully metallised housing
The extensive shielding must be
held tight under the shield fixings -
as a conducting connection with the housing!

Abbildung 1: Shielding in Agreement with EMI

II. Overview

1. Introduction



First of all, congratulations on your mini-size PROCESS-PLC NANO-A!

You will soon realise, that a small device can also be effective and easy to handle.

Programming in plain text high language and a multitasking operating system are a novelty in the category of mini-controllers.

About the Manual

We have also taken new lines in designing the manual hoping that your busy workdays will be seasoned with some humour.

In the appendix you will find a short questionnaire which you may copy and fax back to us. We would appreciate you giving us your opinion about this manual that way, for only you, as the user, can really tell us, whether contents and layout are attractive and refer to everyday experience. We then will consider your ideas for new manuals or for further editions of manuals that are already existing as well as possible.

For the PROCESS-PLC Beginner

For the PROCESS-PLC beginner, we would recommend to read chapter II, where basic characteristics of the PROCESS-PLC technology will be explained. Additionally, programmers who are already very experienced in standard PLC systems will find great help for programming the NANO-A. You will certainly come to appreciate the advantages of process-oriented descriptive programming, as well as the advantages of multitasking.

For PROCESS-PLC Experts

PROCESS-PLC stands for a complete set of controllers

With the PROCESS-PLC NANO-A, the development of the PROCESS-PLC series towards mini-controller is completed. In the NANO-A you will rediscover all features common to you from PROCESS-PLC technology. The philosophy that all PROCESS-PLC systems can be programmed in exactly the same way has also come true without any restrictions in the area of mini-controllers.
This makes the NANO-B control system and its big brother NANO-A unique in their area.
For you as a PROCESS-PLC user this means that you can make use of PROCESS-PLC technology for the whole range of your applications - from simple digital input and output functions, up to complex engineering processes or axis application.

For Everybody

We wish you much fun and success working with the NANO-A PROCESS-PLC.

In case there are any questions or problems, our hotline will be available for you:

Application:	0049-7141 / 2550 - 444
Technical Sales :	0049-7141 / 2550 - 433
e-mail:	jetter@jetter.de

Best Regards

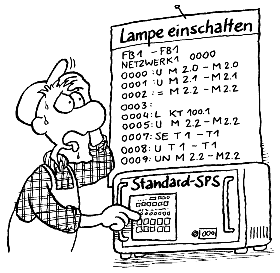
JETTER GmbH

2. PROCESS-PLC: The Technology

2.1 What does PROCESS-PLC Technology Really Mean?

The philosophy of PROCESS-PLC systems can be best explained by a comparison with standard PLC control systems. To highlight the differences it might be good to undertake a journey into the past.

The concept of the standard PLC came into being in the seventies without remarkably changing ever since

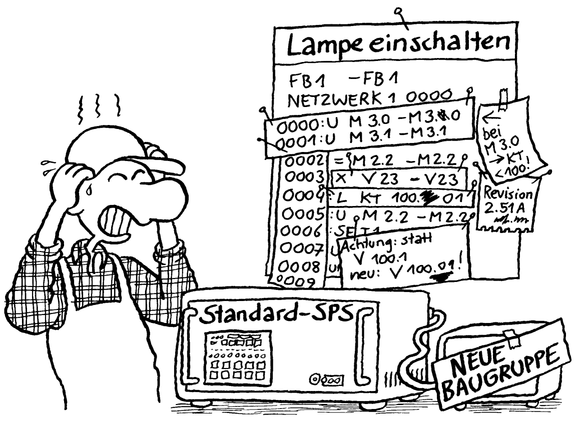


The PLC control system was developed at the beginning of the seventies as a substitute for relays and contactors, the pioneer advantage of flexible programming of functional routines. Programming was carried out in *the ladder diagram* programming language that could be understood by the electrician. *Block diagram and statement list* were soon to complete the language resources that are known today.

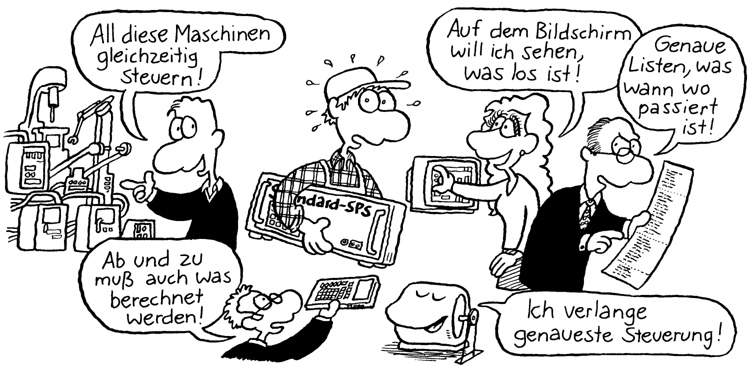
All three languages have got one thing in common: they are closely hardware-oriented. First, only digital and analogue inputs and outputs were required; so, this was no problem. As there are parallel processes in each system, the cyclic storage run was chosen as a processing method.

New Demands on Automation Technique

The world of automation, though, has changed a lot, especially during the last few years. The systems to be controlled are getting more and more complex, thus the demands on automation technique increase.



- Servo- and stepper motor axes
- PID-controllers
- Data management
- Operator guidance
- Process monitoring
- Arithmetic
- Decentralised intelligence
- Operating data acquisition


All diese Maschinen gleichzeitig steuern!
Auf dem Bildschirm will ich sehen, was los ist!
Genaue Listen, was wann wo passiert ist!
Standard-SPS
Ab und zu muß auch was berechnet werden!
Ich verlange genaueste Steuerung!

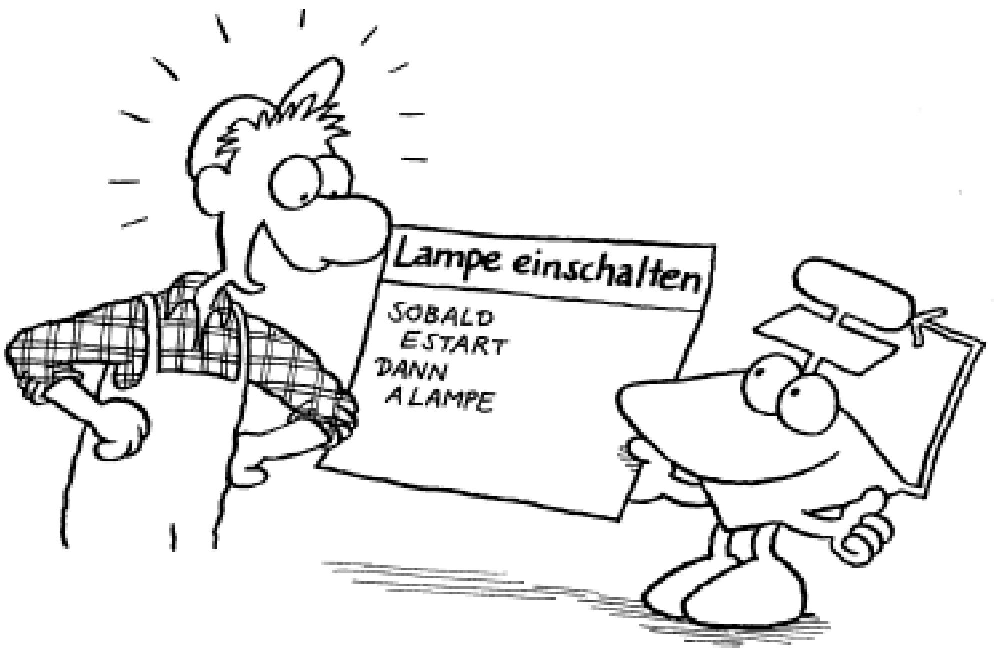
All these functions gain more and more importance.

As the PLC languages ladder diagram, block diagram, and statement list are closely oriented torwards the hardware functions digital input and output, problems can arise here. Many of the required functions can only be realised at great expense. To prevent this, certain functions are transferred to separate devices or assemblies that can be programmed by their own programming languages. This has lead to a great number of software interfaces and to complicated data exchange. User interfaces and positioning routines and functions for example, are programmed separately, with the result of functional restrictions, uneasy programming and long program creation time in consequence.

A new concept for new requirements: PROCESS-PLC

For this reason, JETTER GmbH has totally broken away from this historical concept when developing their PROCESS-PLC technology. The principle is to find new ways in automation technique. The basics of this technology are:

- Direct transfer of the process into a program
- Direct literal description by plain text language programming of the assignment that is to be carried out
- Parallel functions are realised by parallel programming (multitasking)
- All functions can be realised by one single programming language
- No programming expertise will be necessary
- Access to all system parameters in realtime

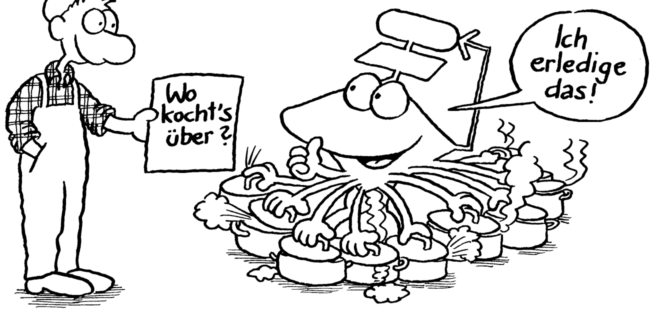


2.2 The Advantage of Multitasking

Multitasking is the logic processing of parallel sequences

In classical PLC technique, the requirements of parallel processing has been realised by cyclic storage run. An actually easier possibility would be multitasking. This expression might remind you of operating systems complex to handle. This is not the case in PROCESS-PLC technology. Using the `TASKBREAK`, `TASKCONTINUE`, and `TASKRESTART` instructions, task control can be realised in an easy way that is clear to understand. Without long initialising routines a program can be written, which can contain up to 32 parallel programs. Parallel functions of the device are structured into definite parallel programs, which are called tasks. By multitasking, subroutine technique, and functions to be parameterised, a clear programming structure can be created.

In classical PLC, functions of various priority are given their individual tasks, while in PROCESS-PLC independent process parts are given their own tasks. That's why they can be easily described, independent of other program parts. This makes the program very easy and clear.



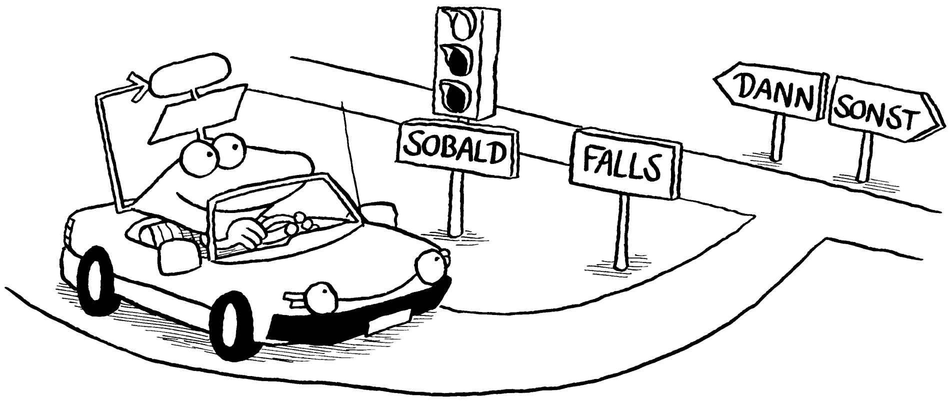
2.3 Process Orientated Versus Cyclic Storage Run

Programming the PROCESS-PLC is directly orientated towards the process; transfer into cyclic processing can be omitted

As parallel processing in PROCESS-PLC technology can be realised by multitasking, cyclic storage run is not necessary!
This, on the other hand, means that the process sequences can be directly transferred into a program run. A short example: An output is to be set at the arrival of an input signal. This is to remain 0,5 seconds to be reset after this. In the SYMPAS plain text language, this can be simply expressed by:

```
TASK 1
        ...
        ...
    WHEN
        I iStart
      THEN
        OUT oValve1
        DELAY 5
       -OUT oValve1
        ...
        ...
```

The WHEN instruction means, that in this task the input signal has to be waited for. This means that the program will not be processed further in cyclic manner. Processing all the other parallel program tasks, though, will be carried out during this time.



If the program flow is to be continued and only a decision to be made, the IF instruction must be used:

```
TASK 1
        ...
        ...
    IF
        OUT oStart
      THEN
        OUT oSlide
      ELSE
        OUT oGripper
        ...
        ...
```

In this example, the start input will be queried at a certain point of the program flow. If it is active at that moment, the output 'slide' will be activated. If the input is not deactivated, the output ‘gripper’ will be set. Other than in the WHEN instruction, the arrival of the condition is not waited for, but merely checked. Depending on the result, one of the two outputs will be set. This means that the program flow need not be interrupted.

By these two examples the machine orientated program flow has been demonstrated. The programmer will not have to transfer the process that has been defined for the machine into a way of thinking according to cyclic processing.

2.4 Plain Text High Level Language SYMPAS

A programming manual is available

SYMPAS programming has been described in a separate manual. For this reason, only some important basics will be explained in this place:

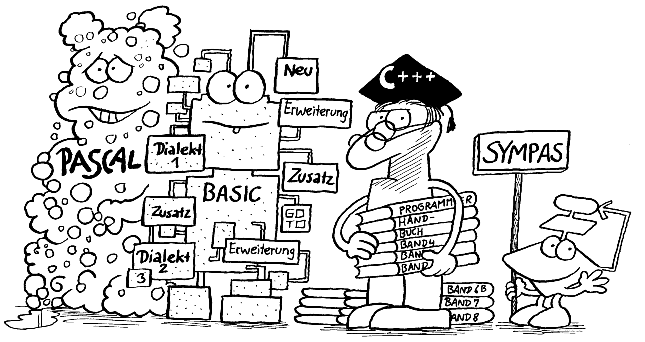
There are two possibilities of beginning to work with the PROCESS-PLC:

1. A basic seminar of three days on programming a PROCESS-PLC
2. Manual: Programming with SYMPAS

SYMPAS: Programming the way you think: Plain text language

The plain text programming language SYMPAS is a high level language adapted to the requirements of automation technique. With this high level language as an excellent basis, a maximum of functional possibilities are provided.
You might ask here, whether none of the famous high level languages like BASIC, C, or PASCAL, are used. The most important reason is the complexity of the known high level languages, which results in the following disadvantages:

1. Long training periods for beginners
1. Problems will be caused in maintaining these programs, as in the standard high level languages a high degree of freedom is allowed
2. Only to be managed by high level language experts



SYMPAS, though, is a programming language accessible by everybody, be it mechanical engineers, electricians, computer scientists, or process engineers - SYMPAS is open to everyone.

SYMPAS: An open programming language accessible by everyone

In SYMPAS, the basic high level language functions and possibilities are made use of; yet, it is less complex than other languages. In addition, SYMPAS has got simple instructions for peripheral functions as, for example, `POSITION` and `DISPLAY_TEXT`.

SYMPAS is a descriptive programming language, close to human thinking. This means, that the deviation via hardware level, that is a characteristic of the standard PLC, can be omitted completely. The process can directly be transferred into a program sequence. The process can directly be transferred into a program run.

SYMPAS can be programmed in German or English language.

All PROCESS-PLC systems are programmed identically

All PROCESS-PLC systems are programmed identically. This means, that programs are portable; thus, a programmer will not have any adjustment problems after changing to another PROCESS-PLC system.

All systems are functioning on a multitasking operating system, which helps to transfer the parallel sequences of each process into parallel programs. This makes the cyclic storage run unnecessary.

In *Chapter II.2.3 Process Orientated Versus Cyclic Storage Run* some instructions of the SYMPAS plain text high level language have already been explained. These are the basic instructions `WHEN` and `IF,` which the program flow is determined by.

SYMPAS: high level language effectiveness without high level complexity

By these instructions, the basis of SYMPAS programming is clearly shown: the high-level language. SYMPAS is a language adapted to automation technique, simple, yet effective.
A problem with the application of standard high level languages in automation is the complexity and the high level of freedom. What is of great effectiveness on one hand, will cause problems with program expansions and maintenance, that is, service, on the other.

SYMPAS has got the central effectiveness of a high-level language, together with the advantage to be generally understood, clar, and easy to manage.

This orientation towards automation technique can be clearly demonstrated by the peripheral instructions:

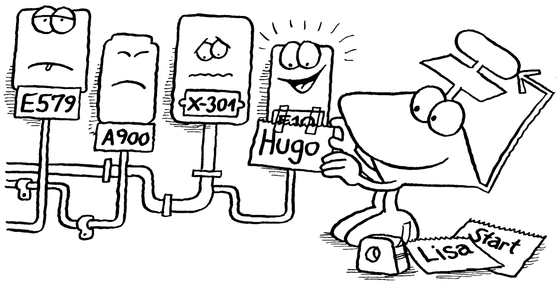
- `DISPLAY_TEXT`
- `DISPLAY_REGISTER`
- `USER_INPUT`
- `POSITION`

By instructions adapted to automation technique programming is made easier

These and further instructions allow easy access to user interfaces, axes, and controllers. This is another characteristic of a high level plain text language: All functions are realised by a programming language and a programming memory. This means, that the texts are not stored in the user interface to be called by the control system, but they are written into the user interface directly out of the operator programming level.

Readability is made easier by symbolic programming

All parameters, like e.g. input and output numbers, analogue values or registers, can be programmed by symbolic
programming



Programming Example Without Symbols:

```
TASK 1
        ...
        ...
    WHEN
        IN 101
      THEN
        OUT 205
```

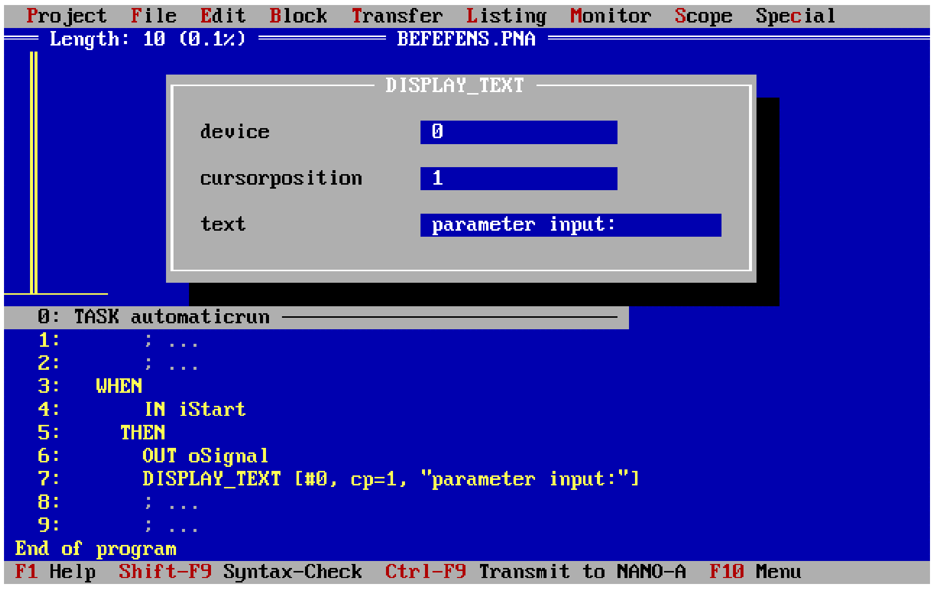
Symbols can either be defined in a symbol file before programming, or directly during programming itself.

Programming Example With Symbols:

```
TASK tAutomatic mode
        ...
        ...
    WHEN
        IN tStart       ;symbolic 101
      THEN
        OUT oSignal     ;symbolic 205
        ...
        ...
```

Easy input of instruction and parametering by input windows

The instructions are parameterised by user friendly input windows and integrated into the program text.


Project File Edit Block Transfer Listing Monitor Scope Special
Length: 10 (0.1%)
BEFEFENS.PNA
DISPLAY_TEXT
F1 Help Shift-F9 Syntax-Check Ctrl-F9 Transmit to NANO-A F10 Menu

```
device          0
cursorposition  1
text            parameter input:
```

```
0: TASK automaticrun
        ; ...
        ; ...
    WHEN
        IN iStart
      THEN
        OUT oSignal
        DISPLAY_TEXT [#0, cp=1, "parameter input:"]
        ; ...
        ; ...
End of program
```

Figure 2: Easy input of instructions by an input window

2.4.1 Overview over Instructions

PROCESS-PLC-Set of Instructions		
Abbr.	**Instruction**	**Remarks**
DR	`DISPLAY_REG`	output of register contents onto LCD or printer
DT	`DISPLAY_TEXT`	output of texts onto LCD or printer
D2	`DISPLAY_TEXT_2`	depending on a register, one of two texts can be chosen
OU	`OUTPUT NUMBER`	setting, resetting, querying of a digital output
U	`USER_INPUT`	input of register values by the user, with the help of the LCD
BC	`BIT_CLEAR`	the bit of a register is cleared or queried for zero
BS	`BIT_SET`	the bit of a register is set or queried for 1
TH	`THEN`	`IF..`**`THEN`**`..ELSE, WHEN..THEN`
DF	`DEF_FUNCTION`	the beginning of a function definition is marked
ED	`END_DEF`	the end of a function definition is marked
IN	`INPUT NUMBER`	a digital input is queried
IF	`IF`	**`IF`**`..THEN..ELSE`
LI	`LIMITS`	1. it is queried, whether the register is inside certain limits (condition) 2. a register is placed between certain limits by force (assignment)
AX	`AXARR`	1. it is queried, whether the axis has been stopped (condition) 2. axis is stopped (assignment)

AP	ACTUAL_POS	the actual axis position is queried
CO	COPY	a register area is copied
NP	NOP	this instruction is of no effect, yet, a processing time is needed (test purposes)
CF	CLEAR_FLAGS	a flag area is cleared
RL	REGISTER_LOAD	a value is written into a register (direct, indirect, doubly indirect)
LA	LABEL	GOTO label for program flow
F	FLAG	setting, resetting, querying a flag
NG	N-GET-REGISTER	a register of a slave control is loaded into the memory of a master control, JETWay, fieldbus
NO	NOT	logic NOT (an input condition is inverted)
NS	N-SEND-REGISTER	a register of a master control is loaded into the memory of the slave control, JETWay, fieldbus
OR	OR	logic OR (input condition)
P	POS	an axis is positioned with speed **v** onto position **pos**
RD	REGDEC	a register value is decremented by 1
RE	REG	register instruction, e.g. REG 100 = 1234
RI	REGINC	a register value is incremented by 1
RC	REG_CLEAR	a register area is set to 0
RZ	REGZERO	a register is set to zero, or a register is queried for zero
RT	RETURN	a subroutine or a function is finished
WH	WHEN	**WHEN**..THEN
SF	specialfunction	call-up of certain special functions, e.g. trigonometry
WM	WHEN_MAX	**WHEN_MAX**..THEN; additionally a time can be input, after which a subroutine (e.g. bugfix) can be called
EL	ELSE	IF..THEN..**ELSE**

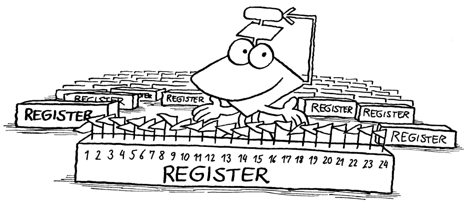
GO	GOTO	control of program flow
ST	START-TIMER	a time register is started
TA	TASK	label for task start
TB	TASKBREAK	a task is breaked
TC	TASKCONTINUE	a breaked task is continued
TR	TASKRESTART	breaked task is started from the beginning
CA	CALL	a subroutine is called up
DE	DELAY	task-processing is breaked for a certain time
WO	WOR	OR linkage of registers
WA	WAND	AND linkage of registers
WX	WXOR	exclusive OR linkage of registers
TE	TIMER-END?	time-register is queried

PROCESS-PLC - Numbers		
Abbr.	**Instruction**	**Remarks**
NB	number (binary)	the numbers are input as binary numbers: b01010101010101010101
ND	number (decimal)	the numbers are input as decimal numbers: 1234
NH	number (hexadecimal)	the numbers are input as hexadecimal numbers: hFA23CD

2.4.2 Data Structure

The NANO-A data are either 24 Bit wide registers or flags (1 Bit).

Registers



260 registers freely disposable to the user

Registers are addresses containing system or user data. The NANO-A control system is equipped with 260 data registers freely disposable to the user, next to various system registers. These can be used for storage of comparative values, results of calculations, measured values or the like.

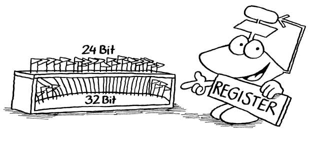
0 .. 199	1000 .. 1059
volatile	non-volatile
user registers	

Register Format

24 Bit registers

All NANO-A registers have got a data format of 24-Bit integer (integer format). This results in a value range from **-8.388.608 to +8.388.607.**

Kinds of Registers



1. **Special registers:** Influencing and monitoring operating system functions, as, for example, task control.
1. **User registers:** Data memories freely disposable to the user.
2. **Peripheral registers**: Registers for peripheral functions, as, for example, user interfaces and analogue inputs / outputs.
4. **Overlapping registers:** Registers, partially overlapped by flags or inputs / outputs.

Flags

Flags can also be classified into flags freely disposable to the user and into special flags, that are used by the operating system or the peripheral functions.

2.5 Complete Access to Functions in Realtime

For the standard PLC several programming languages and several devices to be programmed are needed. Thus, hardware and and software interfaces must be used. Yet, they cause access to various functions like display, axes, and controllers to be complicated and not easy to manage.

PROCESS-PLC: Access to all parameters in realtime

Here, the concept of PROCESS-PLC is of great advantage. As by one programming language all functions can be addressed and realised, access to all function parameters can easily be made in realtime.

Example:

An analogue input value is to be written into the display:

```
TASK 1
   ...
   ...
   DISPLAY_TEXT [#0, cp=1, "temperature:"]
   DISPLAY_REG [#0, cp=14, Reg:rANAIN1]
   ...
   ...
```

First, the text *temperature* is displayed in device 0 (user interface) in cursor position 1. After this text, the contents of register `rANAIN1` (analogue input) is written. This value is accessed directly and output on the user interface.

3. PROCESS-PLC NANO-A: So Small, yet Talking Plain Text

3.1 Basic Device NANO-A

- digital inputs / outputs
- analogue inputs / outputs

Integrated into the Basic Controller:

- 8 digital inputs
- 6 digital outputs
- fast single-channel counter 10kHz
- field bus interface RS485 (JETWay-R)
- interface RS422 of user interface
- programming interface RS 232

Expandability (Basic Device Included):

- 62 digital inputs / outputs
- 4 analogue inputs
- 4 analogue outputs

3.2 Expansion via Internal System Bus

Using the internal system bus, decentralised arrangement of expansion modules is possible

The NANO-A system can be expanded by digital and analogue expansion modules. The internal system bus is a CAN bus. The expansion modules can either be directly coupled with the basic module, or else in decentralised mode in 40 m distance as a maximum from the basic controller. In case of decentralised arrangement, one power supply per decentralised unit will be needed.

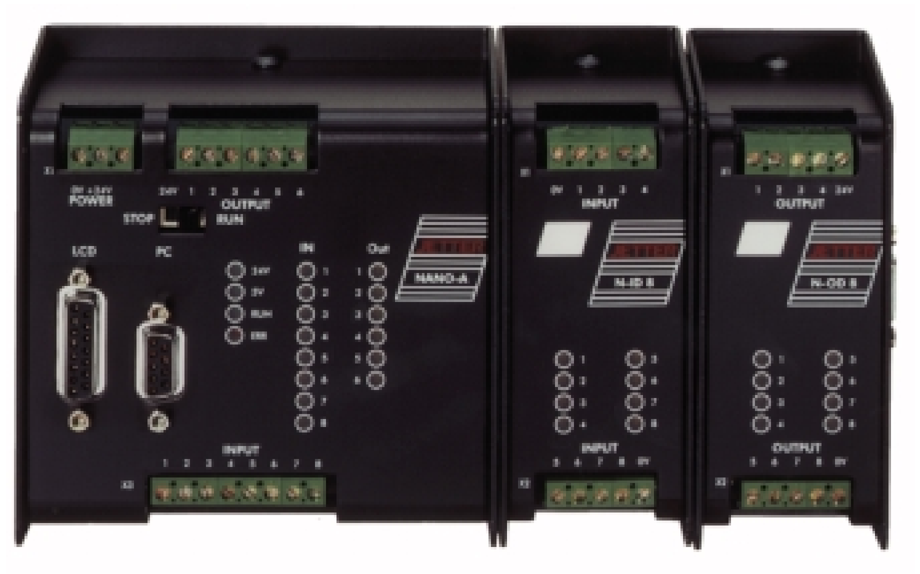


Figure 3: PROCESS-PLC NANO-A with Expansion Modules

3.2.1 System NANO Centralised

For centralised arrangement the expansion modules are directly plugged into the basic controller.

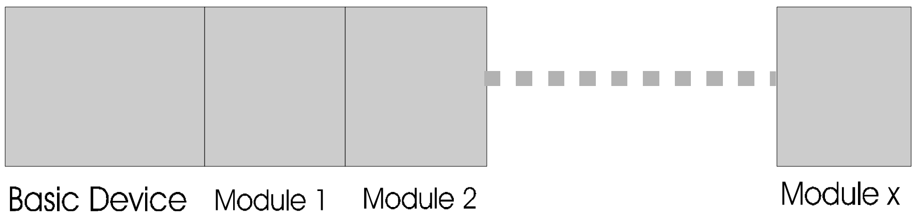


Figure 4: Centralised arrangement of expansion modules. As a maximum, 5 expansion modules are possible.

3.2.2 System NANO Decentralised

By using the CAN-bus as internal system bus, one or more modules can be arranged in up to 40 meters total distance decentralised from the basic controller.

The modules are addressed by the user program, as if they had been arranged in central mode.

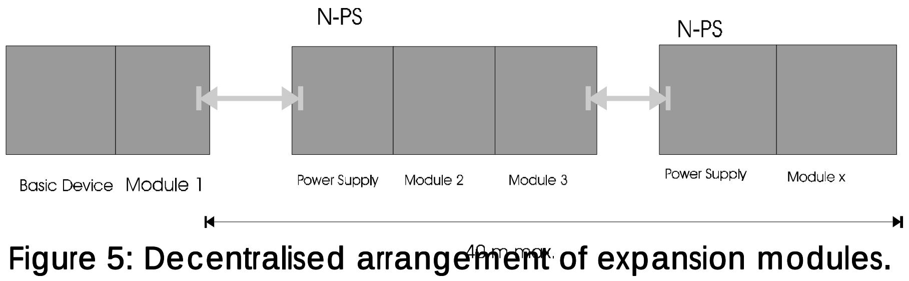


Figure 5: Decentralised arrangement of expansion modules. As a maximum 5 expansion modules are possible. Each decentralised unit must be supplied with an N-PS1 power supply.

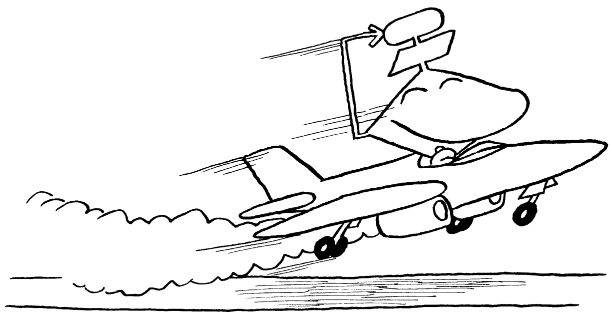
3.2.3 Direct Connection of Festo Valve Blocks to the CAN System Bus

Connection of valve blocks without additional adapters

Valve blocks by Festo of the CP type can be directly connected to the NANO PROCESS-PLC. This means, no special adapter on either valve block or system NANO, will be needed. Connection to the controller is the same as it is with the decentralised arrangement of digital and analogue components.

3.3 Possibilities of Networking

On each of the JETTER control systems, the NANO-A included, there is a JETWay fieldbus interface on the basic unit.



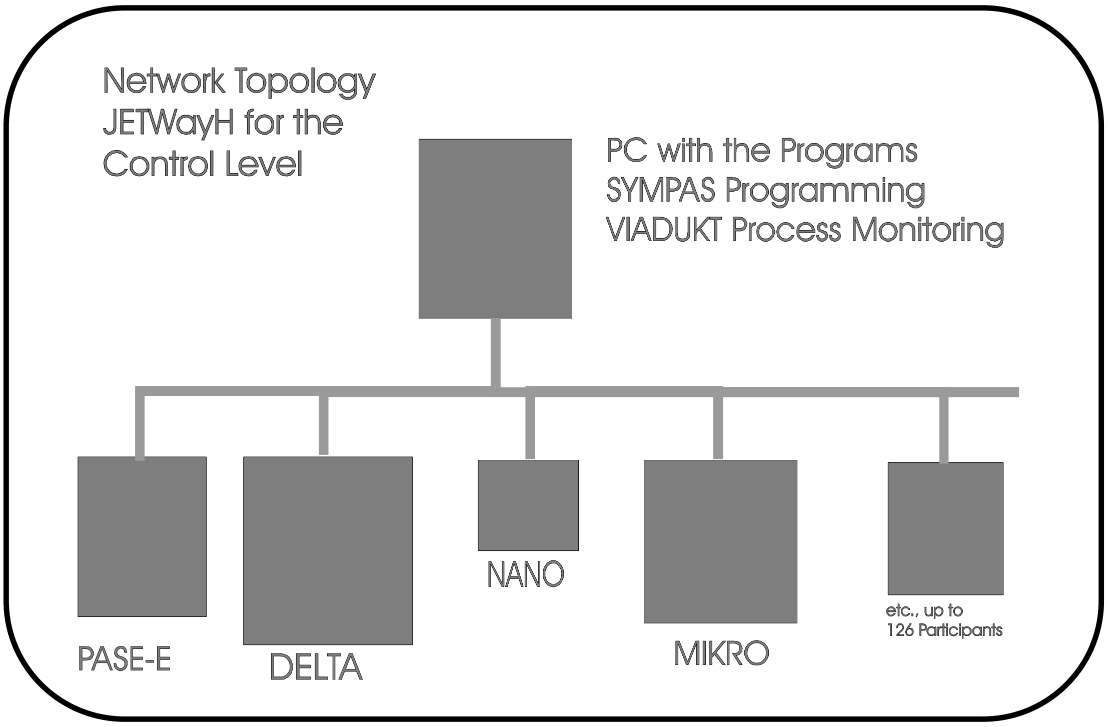
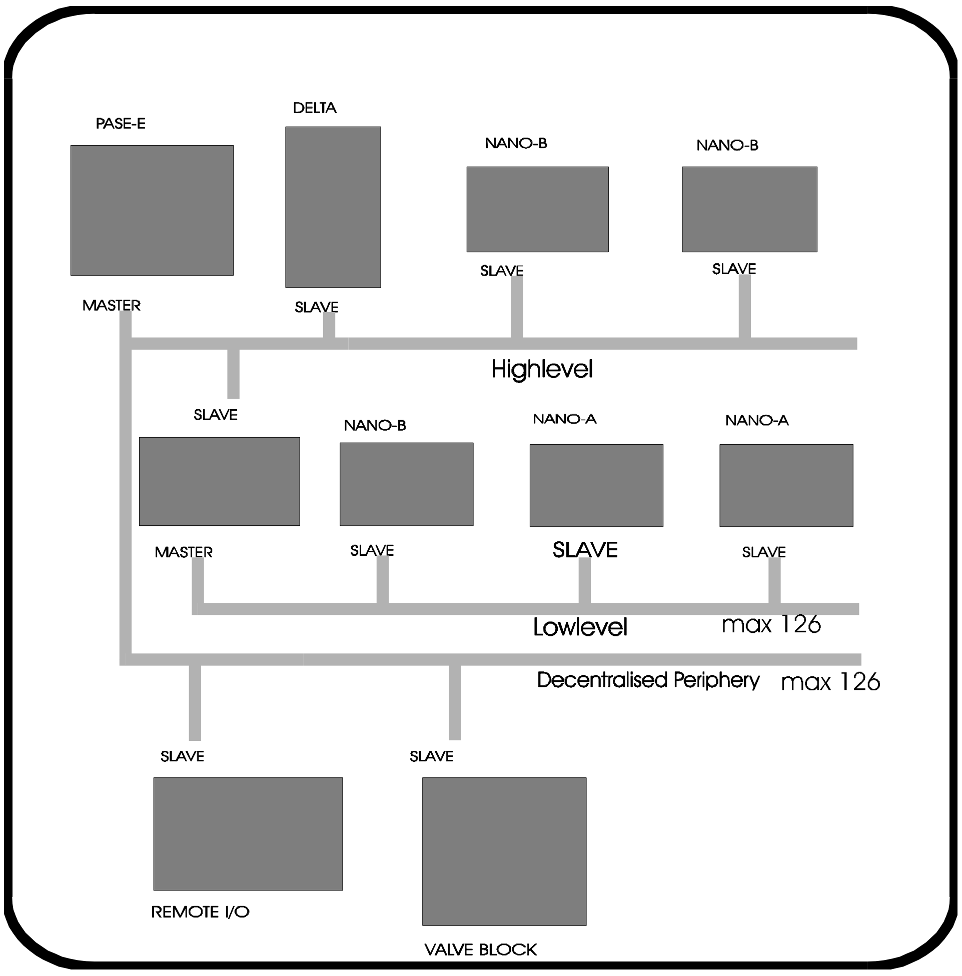


Figure 6: JETWay-H for the Control Level


PASE-E
DELTA
NANO-B
NANO-B
SLAVE
SLAVE
MASTER
SLAVE
Highlevel
SLAVE
NANO-B
NANO-A
NANO-A
MASTER
SLAVE
SLAVE
SLAVE
Lowlevel
max 126
Decentralised Periphery
max 126
SLAVE
SLAVE
REMOTE I/O
VALVE BLOCK

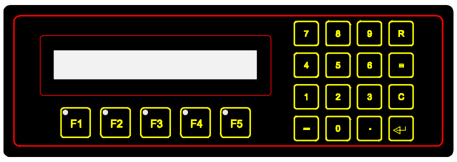
3.4 User Interfaces, Process Monitoring

The user interfaces are also controlled directly by the user program; they need not be programmed separately

Various user interfaces have been supplied for operator guidance. If still more complex processes are to be displayed, one can choose between graphic data processing devices and the PC compatible monitoring system VIADUKT, which contains additional functions for operating data acquisition and data management.

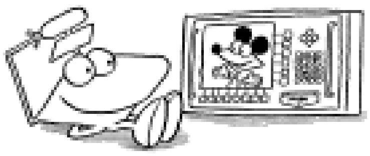
Pictures of Devices

LCD 17

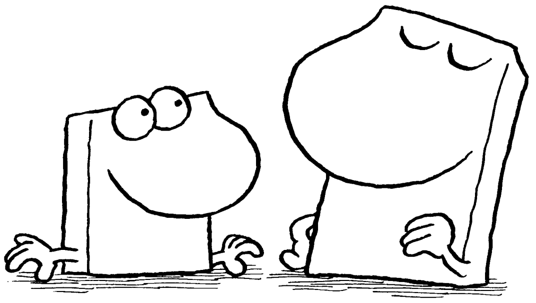


LCD 34

VIADUKT



3.5 Big Brother NANO-B



Big brother NANO-B for digital and analogue inputs, outputs, axes, PID controllers, and a lot more

The following functions can be carried out by the NANO-B basic controller:

- 8 digital inputs 24 V =
- 8 digital outputs 24 V =, 0.5 A
- stepper motor control (up to 5 kHz)
- 4 analogue inputs 8-Bit
- 1 analogue output 8-Bit
- Fast single channel counter 10 kHz
- Fast dual channel counter 5 kHz
- Programming interface RS232
- Fieldbus interface RS485
- Interface for operating devices RS232 / RS422
- Freely programmable interface RS232 / RS422 / RS485
- Realtime clock
- System bus interface for centralised or decentralised expansions

To be expanded up to (basic device included):

- 136 digital inputs / outputs
- 28 analogue inputs
- 13 analogue outputs
- 3 servo axes
- 7 stepper motor axes
- 12 PID controllers
- 4 hardware counters

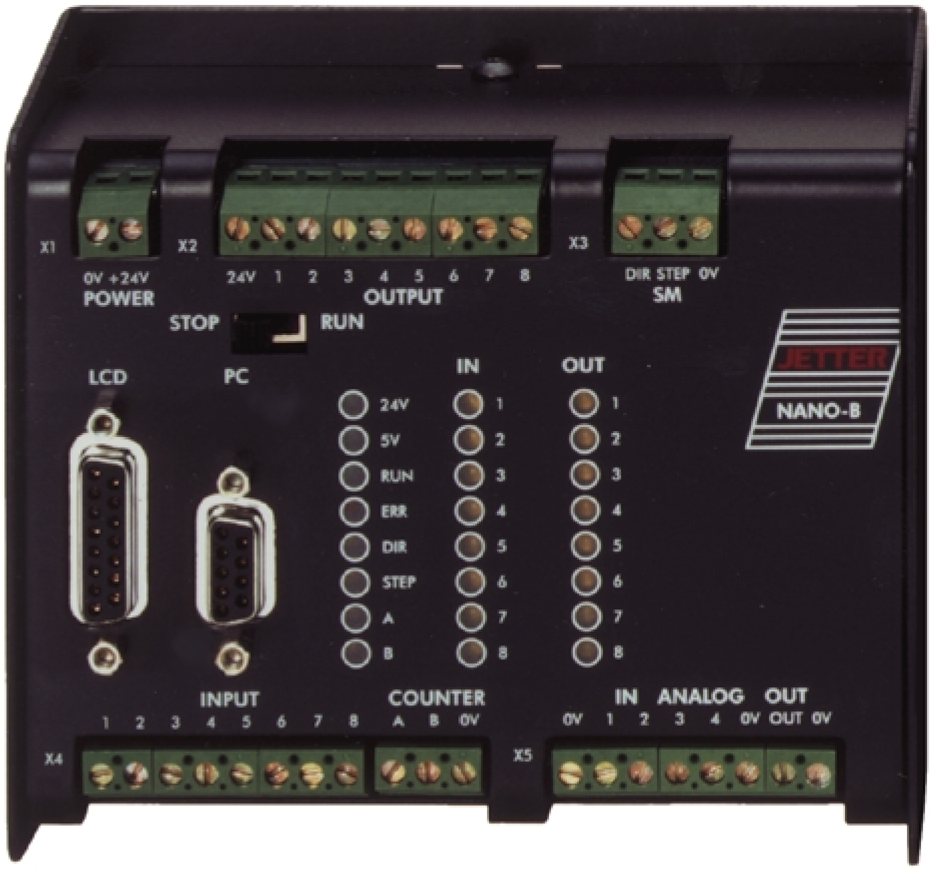


Figure 6: PROZESS-PLC NANO-B without Expansion Modules

III. Basic Device NANO-A (Hardware)

1. Overview and Technical Data

Basic Device NANO-A	
Program memory	6 kByte EEPROM
User register 24 Bit	250 (200 volatile; 50 EEPROM)
Data format	24 Bit Integer: - 8.388.608 ... + 8.388.607
Internal intermediate results	32 Bit
Number of user flags	256
Digital inputs	8 (24 VDC)
Digital outputs	6 (24 VDC, 0.5 A)
Fast single channel counter	10 kHz (24V)
Programming interface	RS 232
Operator and monitoring interface	RS 232 / RS 422
Field bus interface JETWay	RS485
System bus interface	CAN-bus interface
Dimensions (H x W x D in mm)	110 x 114 x 70
Mass	600g
Mounting	DIN rail

Connections	
Voltage supply	Terminal screws
Digital inputs / outputs	Terminal screws
Programming interface	9-pin SUB-D female connector
JETWay field bus interface	15-pin SUB-D female connector
System bus interface	9-pin SUB-D female

	connector
Operator and monitoring interface	9-pin SUB-D female connector 15-pin SUB-D female connector

Voltage Supply	
Power consumption, incl. 8 digital outputs, except expansion modules	ca. 25 Watt
Power consumption for centralised, fully equipped configuration (without power supply for decentralised arrangement)	ca. 100 Watt
Demands on power supply	24VDC, -15% - +20%

Switches, LED's	
STOP/RUN switch	When the switch is set to STOP position, the user program will not be started after attaching the voltage supply
LED 24 V	Voltage supply 24V o.k.
LED 5 V	Internal logic supply o.k.
LED RUN	User program is running
LED ERROR	Error in register 2008 - 2012
LED IN 1 - 8	24V input feed
LED OUT 1 - 6	Output set on 24V

2. Mechanical Dimensions

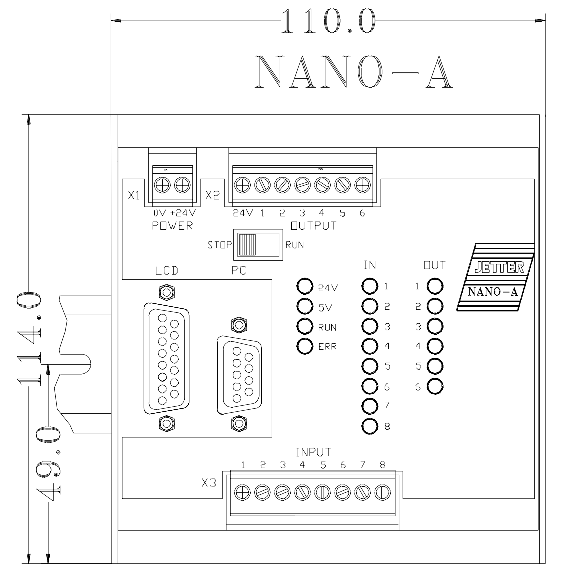
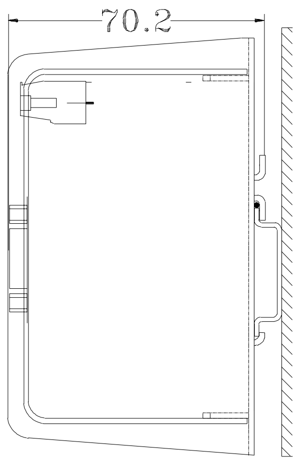


Figure 7: Mechanical Dimensions of the NANO-A Basic Device



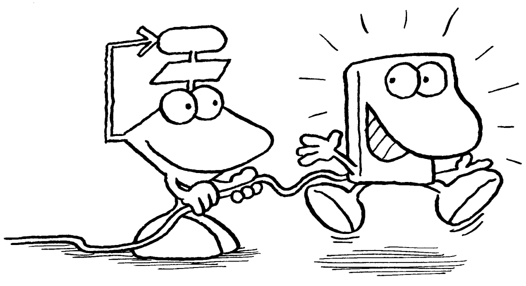
Note:

The depth shown here refers to all NANO modules.

Figure 8: The Side View of the NANO-A Modules

3. Electrical Connections, Specifications

3.1 Power Supply



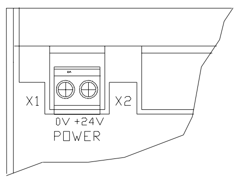


Figure 9: Connection of Power Supply

24 V power supply with the following characteristics:

Voltage range:	20 V .... 30 V
Filtered:	Remaining ripple 5 %
Power:	ca. 50 W for completion

In case of centralised arrangement the digital expansion modules are also supplied by the basic controller. In case of decentralised arrangement power is supplied by the digital expansion modules via the N-PS power supply (see *Chapter IV. 4. The N-PS1 Module, Power Supply for Decentralised Modules*). The intelligent expansion modules have got their own connection for the 24 V power supply.

3.2 Interfaces

On the basic device there are three female connectors for the various interfaces.

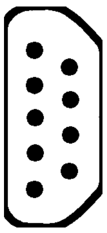
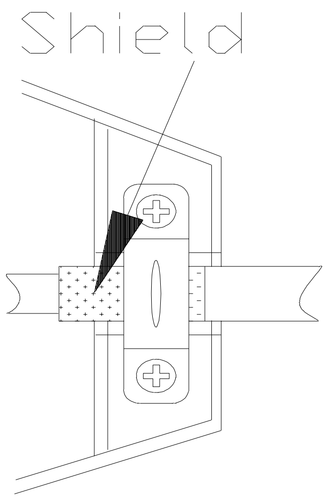
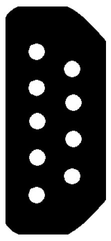
Interfaces of the Basic Controller		
Interface	**Function**	**Specification**
9 pin SUB-D (front)	programming monitoring JETWay-H, -R	RS232 RS232 RS485
15 pin SUB-D (front)	programming operating devices monitoring JETWay-R	RS232 RS422 RS232 RS485
9 pin SUB-D	expansion by modules connected to system bus	

Simultaneous use of the following interfaces is not possible:

Restrictions in case of Simultaneous Use of Various Interfaces		
Interface	**Simultaneous use is not possible**	**Simultaneous use is not possible**
RS232 (9 pin)		
RS232 (15 pin)		not
RS485 (9 pin)	not	
RS485 (15 pin)	not	
RS422 (15 pin)		not

3.2.1 Programming Interface to PC (RS232)

See also *chapter III, 3.2.2 Programming Interface to the PC (JETWay-H)*

Programming Cable (EM-PK)			
PROCESS-PLC			PC
9 pin, SUB-D female connection 	RS232 max. cable length: 15m  **Please shield extensively! Only use metallised housings!**		9 pin SUB-D female connection 
Pin	Signal		Pin
2	TxD	RxD	2
3	RxD	TxD	3
7	Gnd		5
On the PC-side (COM1), pins 7 and 8 and pins 1, 4, and 6 are to be bridged.			

Note:

The connection cable EM-PK can be obtained from JETTER.
If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	3
Diameter:	$0,25^2$
Connection:	SUB-D male metallised
Shielding:	total, not paired

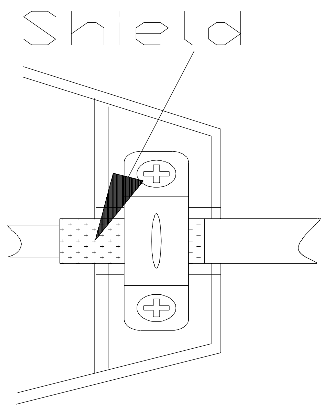
The shield must on both sides have extensive contact to the connector shells.

3.2.2 Programming PC Interface (JETWay-H)

JETWay-H:
126 participants
115 kBaud

Using the JETWay-H interface as a programming interface is more advantageous compared to using the RS232 interface:

- Up to 126 PROCESS-PLC can be accessed from one SYMPAS workstation
- Transfer data of up to 115kBaud can be realised.

JETWay-H Cable		
Connection on the NANO-A Side	Shielding	Specification max. Length
9 pin male SUB-D connector or 15 pin male SUB-D connector	 **Please shield extensively! Only use metallised housings!**	RS485 max. cable length: 400m
Pin	Signal	Remarks
7	Gnd	
8	Data +	
9	Data -	

The JETWay-H Board for the PC

With the help of the PC board shown below, the connection between SYMPAS and up to 126 PROCESS-PLC control systems via JETWay-H can be realised.

-------------------→

By these DIL switches, the port address can be defined.
The default is 340h..

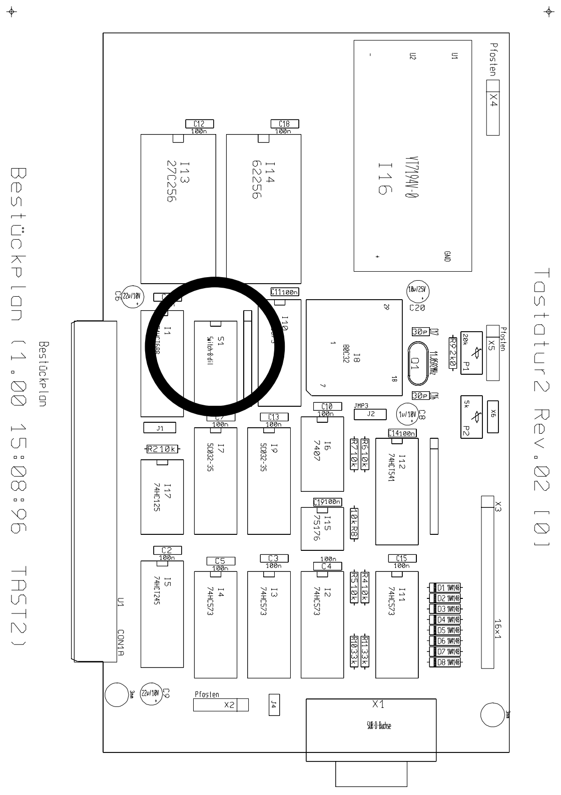


Figure 10: JETWay-H-board for the PC

AUTOEXEC.BAT

In the AUTOEXEC.BAT of your PC the following line must be inserted (on the condition that default has been set):

```
SET JETWAY_PORT=340h
```

DIL SWITCH

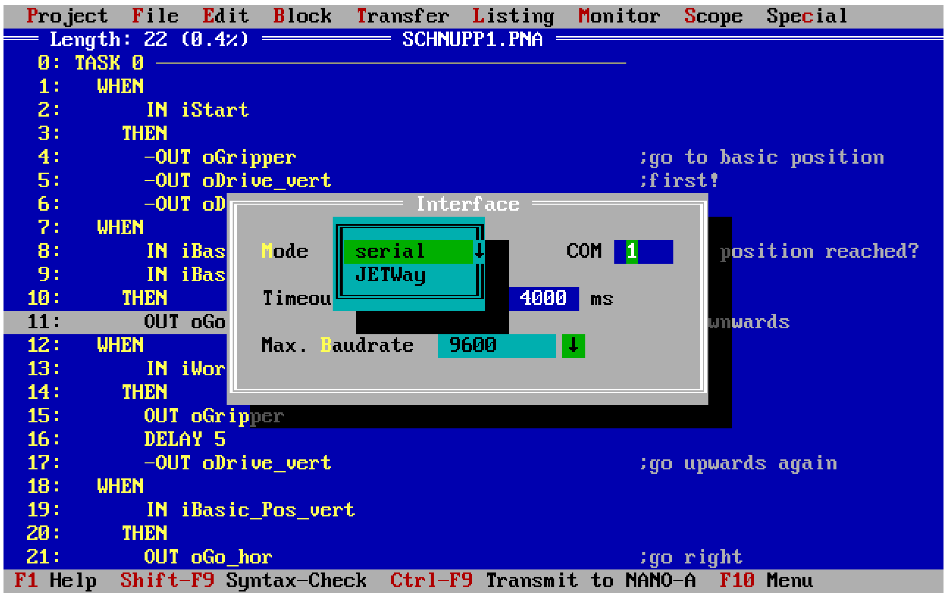
A different port address can be chosen using the DIL switches on the JETWay-H board as shown above.

DIL Switches on the JETWay-H Board						
Port	Switch 7	Switch 6	Switch 5	Switch 4	Switch 3	Switch 2
300h	OFF	OFF	ON	ON	ON	ON
310h	OFF	OFF	ON	ON	ON	OFF
320h	OFF	OFF	ON	ON	OFF	ON
330h	OFF	OFF	ON	ON	OFF	OFF
340h*)	OFF	OFF	ON	OFF	ON	ON
350h	OFF	OFF	ON	OFF	ON	OFF
360h	OFF	OFF	ON	OFF	OFF	ON
*) Default setting						

Correspondingly, the line in AUTOEXEC.BAT has to be changed:

```
SET JETWAY_PORT=x
```

In the SYMPAS menu “Special / Interface“ a choice can be made between a programming interface via RS232 or via JETWay-H.



```
  Project  File  Edit  Block  Transfer  Listing  Monitor  Scope  Special
══ Length: 22 (0.4%) ═══════════ SCHNUPP1.PNA ══════════════════════════════
 TASK 0 ────────────────────────────────────────
   WHEN
       IN iStart
     THEN
       -OUT oGripper                                     ;go to basic position
       -OUT oDrive_vert                                  ;first!
       -OUT oD ══════════ Interface ══════
   WHEN
       IN iBas    Mode     serial ↓          COM 1       position reached?
       IN iBas             JETWay
     THEN         Timeou            4000  ms
       OUT oGo                                          wnwards
   WHEN           Max. Baudrate  9600  ↓
       IN iWor
     THEN
       OUT oGripper
       DELAY 5
       -OUT oDrive_vert                                  ;go upwards again
   WHEN
       IN iBasic_Pos_vert
     THEN
       OUT oGo_hor                                       ;go right
 F1 Help  Shift-F9 Syntax-Check  Ctrl-F9 Transmit to NANO-A  F10 Menu
```

Figure 11: SYMPAS Menu: Special / Interface

Note:

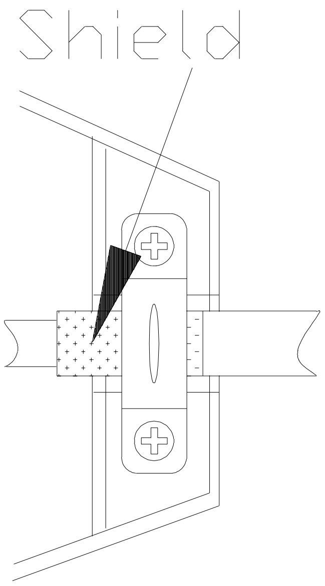
If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	3
Diameter:	0,25^2
Connection:	SUB-D, metallised
Shielding:	total, not paired

The shield must on both sides have extensive contact with the connector shells.

3.2.3 Network Interface (JETWay-R)

JETWay-R serves for networking PROCESS-PLC's and/or networking REMOTE I/Os, valve blocks, etc. with the PROCESS-PLC. See *Chapter VII. Network Operation.*

JETWay-R Cable		
Connection on the NANO-B	Shielding	Specification maximum length
9 pin SUB-D male connector or 15 pin SUB-D male connector	 **Please shield extensively ! Only use metallised housings!**	RS485 max. cable length: 400m
Pin	Signal	Remarks
7	Gnd	
8	Data +	
9	Data -	

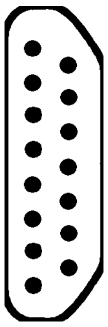
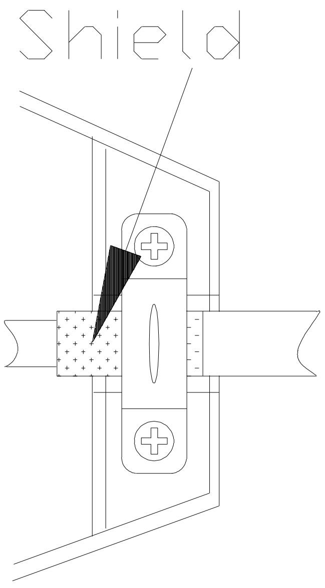
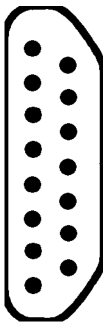
Note:

If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	3
Diameter:	$0,25^2$
Connection:	SUB-D male, metallised
Shielding:	total, not paired

The shield must on both sides have extensive contact with the connector shells.

3.2.4 User Interface Connections

User Interface Cable (DK-422)			
PROCESS-PLC			User Interface
15 pin male SUB-D connector 	RS422 max. cable length: 400m  **Please shield extensively!** **Only use metallised housings!**		15 pin male SUB-D connector 
Pin	Signal		Pin
4	24 VDC		15
7	Gnd		12
10	SDB	RDB	6
11	SDA	RDA	7
12	RDB	SDB	4
13	RDA	SDA	5

Note:

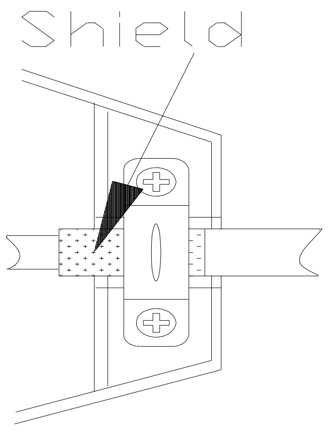
The prefabricated connection cable DK-422 incl. male connector for operating devices can be obtained from JETTER.
If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	8
Diameter:	$0,25^2$
Connection:	SUB-D male, metallised
Shielding:	total, not paired

The shield must on both sides have extensive contact with the connector shells.

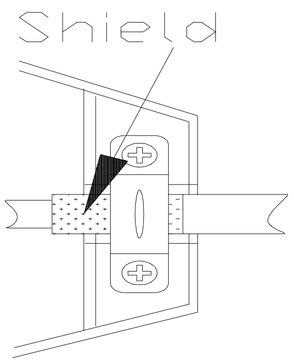
3.2.5 Monitoring Interface

The VIADUKT process monitoring system can be connected with the PROCESS-PLC by two different cables.

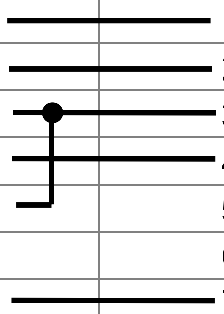
VIADUKT Cable		
Connection		VIADUKT
9 pin SUB-D male connector or 15 pin SUB-D male connector	 **Please shield extensively! Only use metallised housings!**	RS232 max. cable length: 15m
Pin	Signal	Remarks
2	TxD	2
3	RxD	3
7	Gnd	5

3.2.6 System Bus for Expansion Modules

On the right hand side of the basic controller respectively the expansion modules the connection for the expansion modules has been placed. In centralised arrangement, they are plugged into the basic controller directly; in decentralised arrangement via a connection cable.
A detailed description of the CAN bus and of the expansion modules will be given in *chapter IV. 1. Basics on CAN System Bus Topology.*

System Bus for Expansion Modules		
Connection	Shielding	Specification max. length
9 pin SUB-D male respectively female connector	 **Please shield extensively! Only use metallised housings!**	CAN max. cable length: 40m

Signal	Pin	Pin (female connector)
CMODE0	1	1
CANL	2	2
Gnd	3	3
CMODE1	4	4
TERM	5	5
free	6	6
CANH	7	7
free	8	8
5 VDC	9	9



Specification System Bus for Expansion	
Transfer rate	1 MBits/s
max. bus length	40m

Note:

The connection cables incl. male connector for decentralised arrangement can be obtained from JETTER.
If you make the cables yourself, the following minimum requirements must be considered:

Number of wires:	7
Diameter:	$0,25^2$
Connection:	SUB-D male, metallised
Shielding:	total, not paired

The shield must on both sides have extensive contact with the connector shells.

3.3 Digital Inputs

On the lower side of the basic controller, eight terminal points have been provided for the 24V input signal. The 0V signal is connected to the control cabinet ground (Gnd).

Technical Data of Inputs	
Number of inputs	8
Nominal input voltage	24 VDC
Voltage range	15 .. 27 V
Input current	ca. 8 mA
Input resistance	3,0 kΩ
Input delay	ca. 3ms
Signal voltage ON	min. 15 V
Signal voltage OFF	max. 10 V
Potential insulation	none

Numbering of the Basic Controller Inputs	
Input	**Number**
Input 1	101
Input 2	102
...	...
Input 8	108

Also see *Chapter V. 1. Addressing of the Digital Inputs and Outputs*.

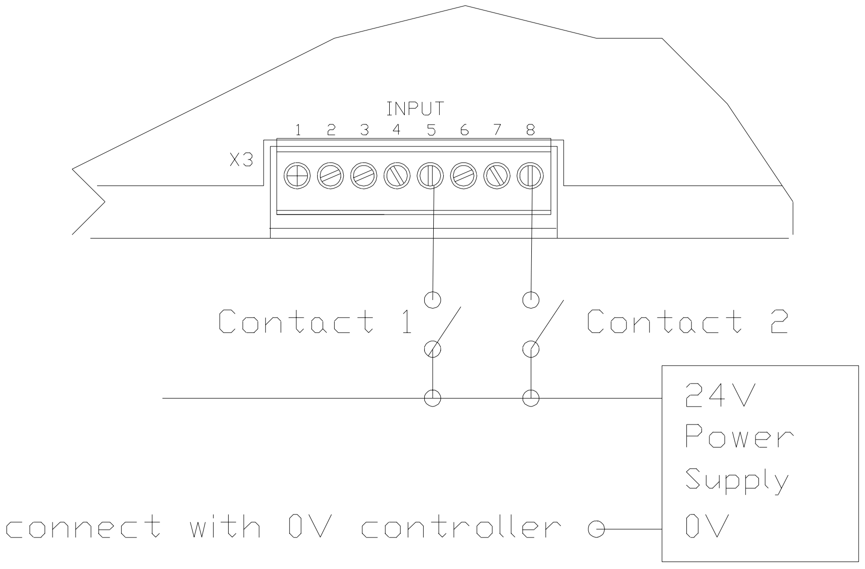


Figure12: Wiring of the digital inputs

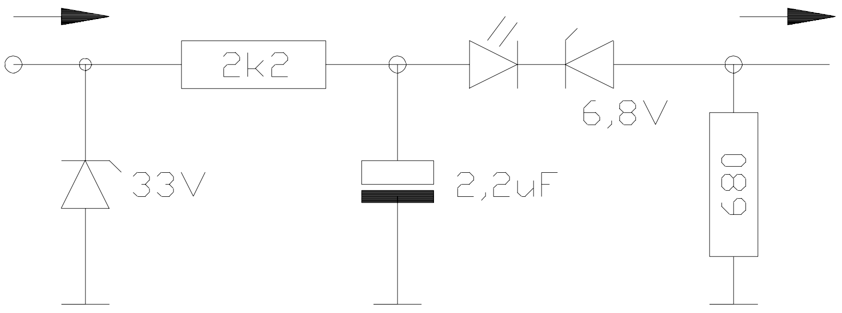


Figure 13: Internal circuit of the digital inputs

3.4 Digital Outputs

The outputs are positioned on the upper six terminal screws provided for this purpose. The 0V signal is connected to the control cabinet ground (Gnd).

Technical Data Outputs	
Number of outputs	6
Kind of outputs	Transistor, pnp
Nominal voltage	24 VDC
Voltage range	20 .. 30 V
Load current	max. 0,5 A / output
Potential isolation	none
Protection switch	Overload, over-voltage, over-temperature
Protection from inductive loads	yes
Signal voltage ON	type. $V_{Power Supply}$ - 1,5 V

Numbering of Outputs on the Basic Controller	
Output	**Number**
Output 1	101
Output 2	102
...	...
Output 6	106

See also *Chapter V. 1. Addressing of the Digital Inputs and Outputs*

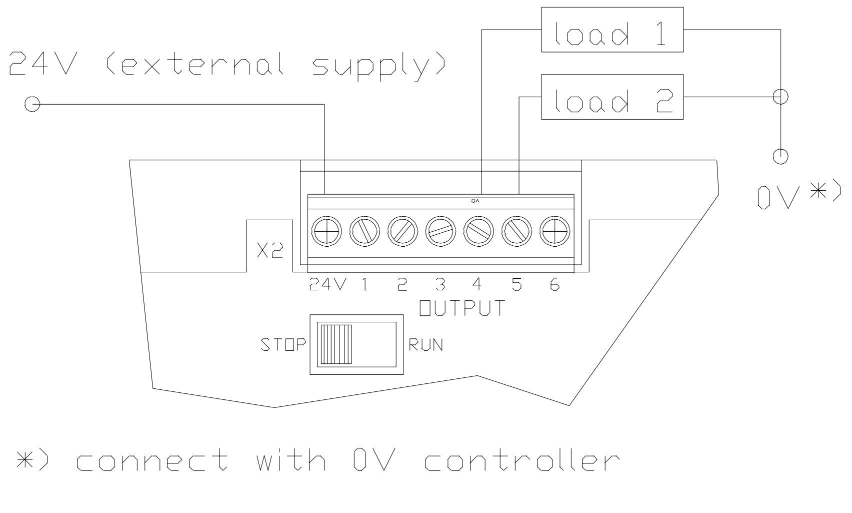


Figure 14: Connection of Digital Outputs

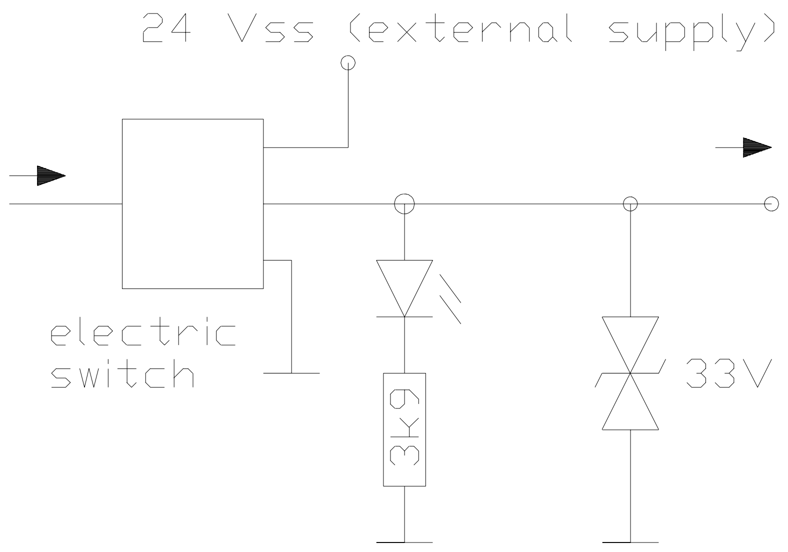


Figure 22: Internal Circuit of Digital Outputs

3.5 Single Channel Counter

Events of a frequency of up to 10kHz can be evaluated by the single channel counter. For this purpose the digital input INPUT 1 is connected.

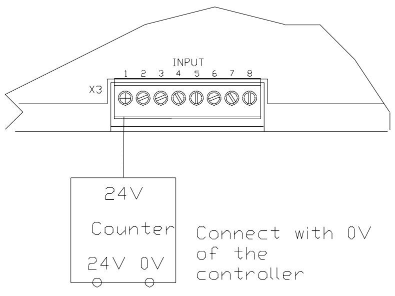
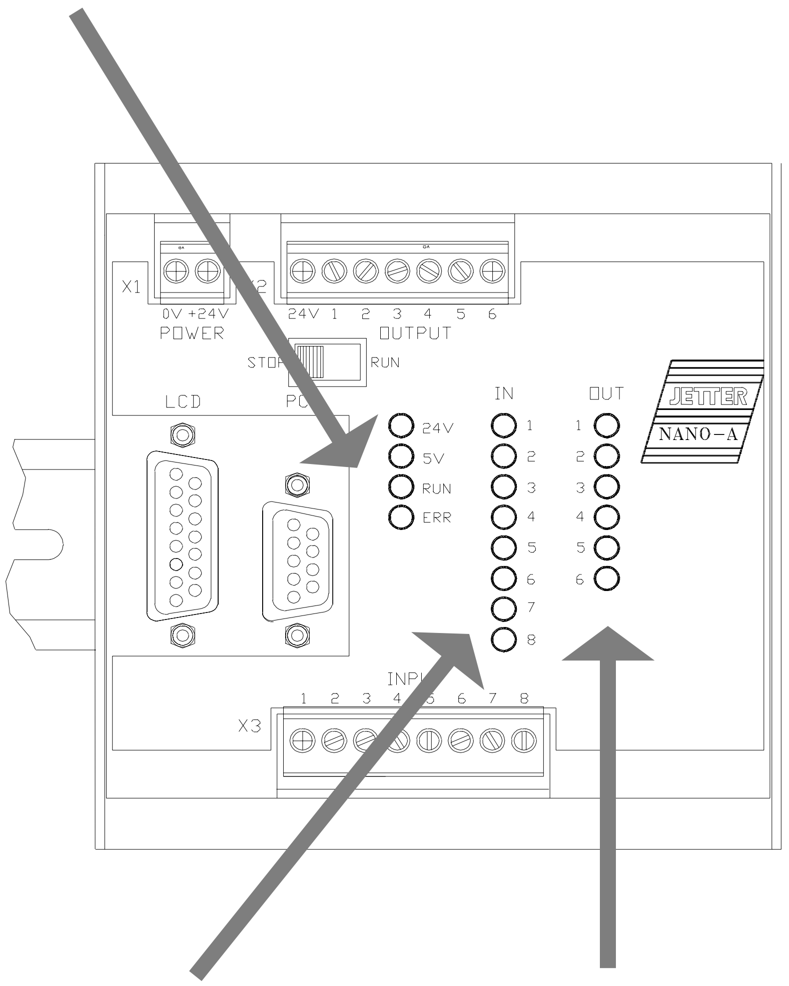


Figure 15: Connection of Single Channel Counter

4. Description of the LED's

24V	Operating voltage OK	
5V	Internal logic voltage OK	
RUN	*lit:*	User program is running
	flashing:	User program is not running Switch set on "Stop"
ERR	Error. The error state has been specified in registers 2008 to 2012	



Digital
Input is Active
24V signal is connected

Digital
Output is Active
24V signal is activated

5. Description of the Mode Switch

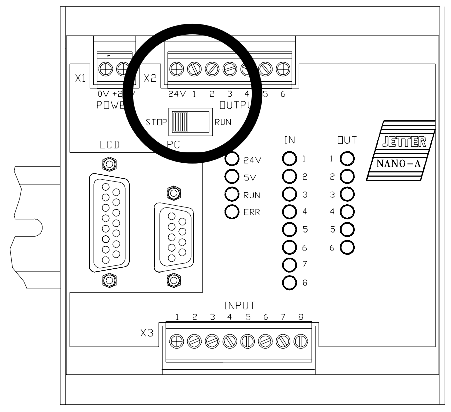


Figure 16: The STOP-RUN Switch

STOP Position

If, at the point of attaching the voltage supply to the control system, the switch is in STOP position, the user's program will not start. It can be activated by pressing Shift-F2 in the SYMPAS program.

RUN Position

If, at the point of attaching the voltage supply to the control system, the switch is in RUN position, the user's program will start.

IV. Expansion Modules

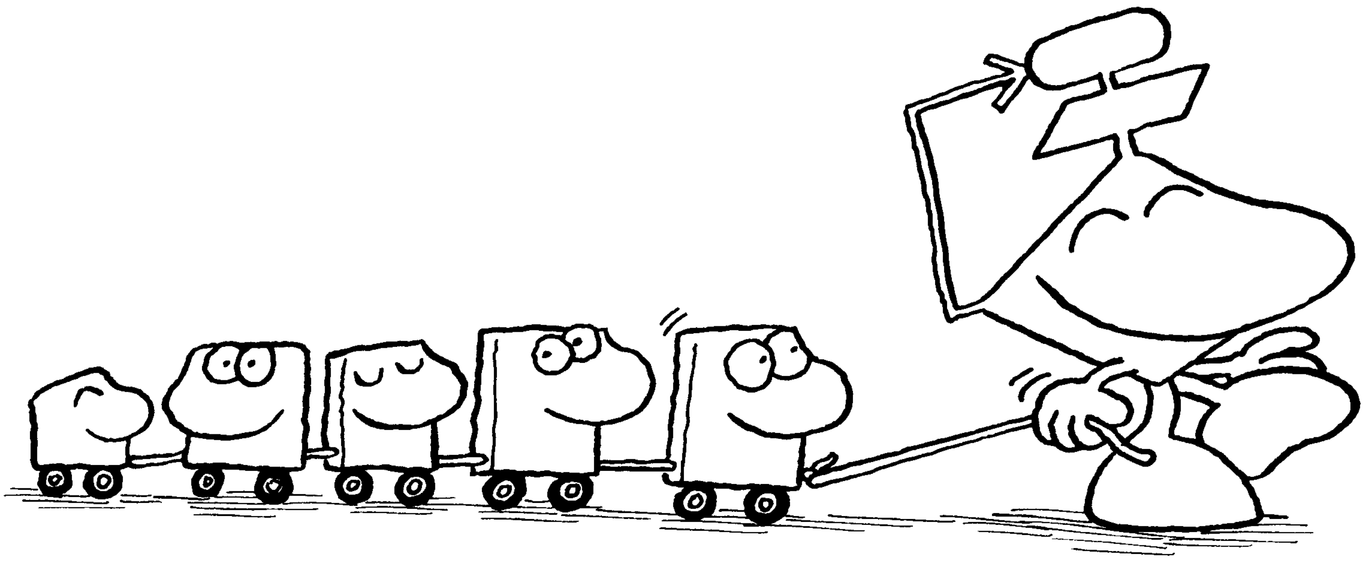
1. Basic Remarks on the CAN System Bus Topology

Thanks to the internal system bus, decentralised arrangement of expansion modules is possible.

The NANO-A operating system can be expanded using additional digital and analogue modules. The internal system bus is a CAN-bus. The expansion modules can either be directly connected to the basic module, or else be placed in decentralised position 40 meters distant as the most from the basic controller. In case of decentralised arrangement, one power supply per decentralised unit will be needed.

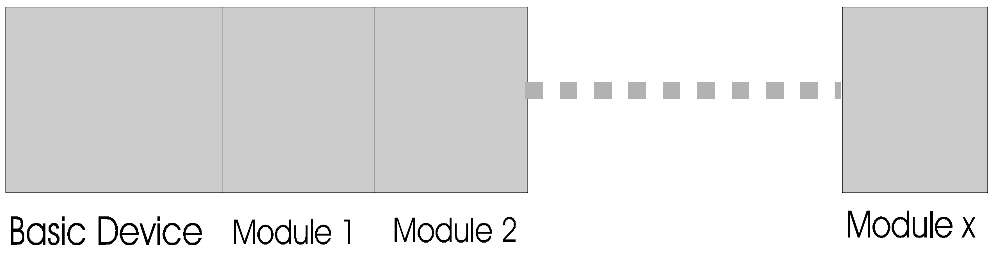
To be expanded to (basic device included)

- 62 digital in-/outputs
- 4 analogue inputs
- 4 analogue outputs



1.1 Centralised Arrangement at the CAN System Bus

In case of centralised arrangement the expansion modules are directly plugged into the basic controller by a mechanical SUB-D connection. Its advantage is the reliability of mechanical and electrical functions as well as good EMI characteristics.



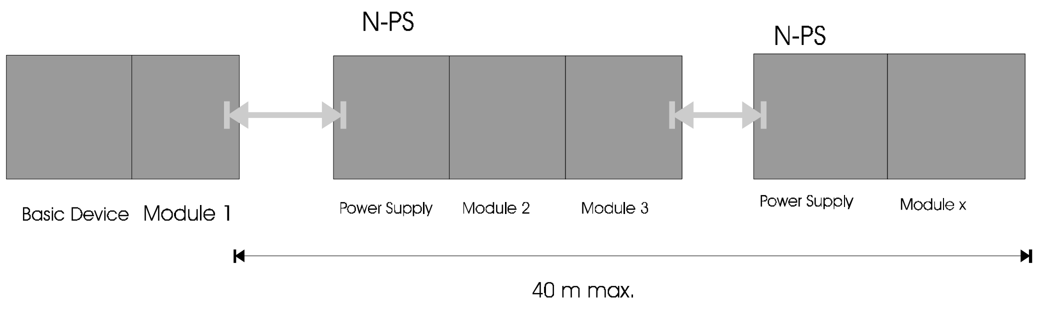
1.2 Decentralised Arrangement at the CAN System Bus

In case of decentralised arrangement, a N-PS1 power supply per unit will be needed.

By using the CAN bus as an internal system bus, one or more than one modules can be placed in a total distance of up to 40 m from the basic controller. Only the N-PS1 power supply module will be needed. The modules are being accessed by the user's program, as if they were centralised.

Note:

In case of decentralised arrangement at the CAN system-bus, a N-PS1 power supply per unit is connected to the central device.



2. The N-ID8 Module, 8 Digital Inputs

2.1 Overview and Technical Data

Overview: Module N-ID8	
Digital inputs	24 VDC -15% .. +20%
Voltage supply	centralised arrangement: by basic controller decentralised arrangement: by N-PS1 power supply
Connection to basic controller by system bus	SUB-D male connection, 9 pin
Connection of inputs:	terminal screws
LED inputs 1 - 8	24 V have been attached to the input
Dimensions (H x W x D in mm)	114 x 45 x 70
Weight	350 g
Mounting	DIN rail

Technical Data of Inputs	
Number of inputs	8
Rated input voltage	24 VDC -15% .. +20%
Voltage range	0 .. 30 V
Input current	ca. 8 mA
Input resistance	3,0 kΩ
Input delay	ca. 3 ms
Signal voltage ON	min. 15 V
Signal voltage OFF	max. 10 V
Potential isolation	none

2.2 Mechanical Dimensions

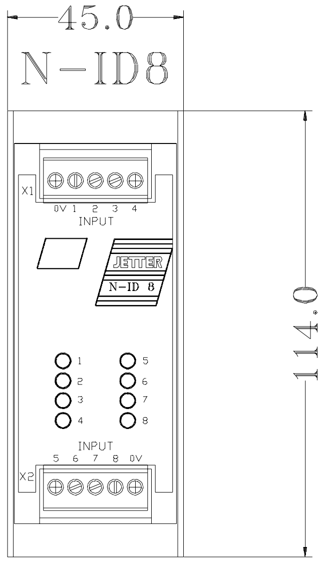


Figure 17: Mechanical Dimensions of the Digital Input Module

2.3 Description of Connections

For the inputs, there are eight terminal points for the 24 V signal available at the expansion device. The 0V signal is connected to the control cabinet ground (Gnd).

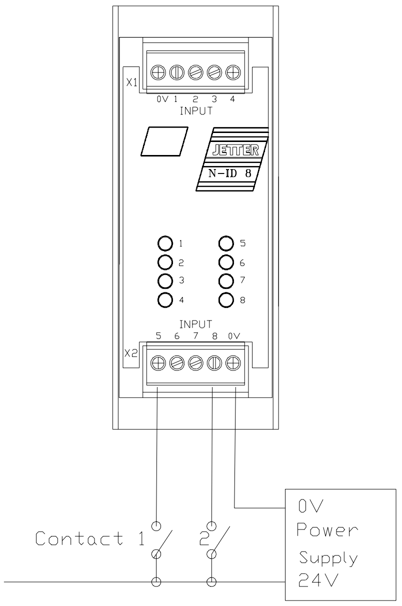


Figure 18: Exemplary Input Wiring of an N-ID8 Module

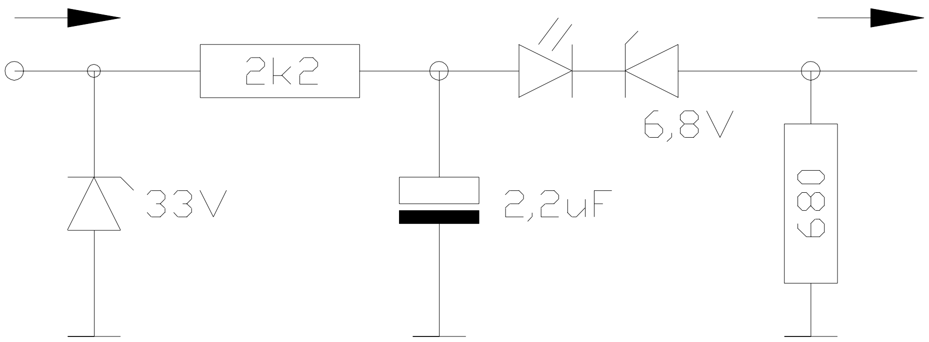


Figure 19: Internal Circuit of the Digital Inputs

Accessing the Digital Inputs

On accessing the digital inputs, see *chapter V.1 Accessing the Digital Inputs and Outputs.*

2.4 Description of the LED’s

The LED’s indicate that a 24V input signal has been activated on the corresponding input.

3. The N-OD8 Module, 8 Digital Outputs

3.1 Overview and Technical Data

Overview: Module N-OD8	
Digital outputs	Transistor 24 V =, 0.5 A
Voltage supply, internal logic	Centralised arrangement: by basic controller. Decentralised arrangement: by N-PS1 voltage supply
Connection to basic controller via system bus	SUB-D male connector, 9 pin
Connection of outputs:	Terminal screws
LED outputs 1 - 8	24 V output set
Dimensions (H x W x D in mm)	114 x 45 x 70
Weight	350g
Mounting	DIN rail

Technical Data of Outputs	
Number of outputs	8
Output types	Transistor, pnp
Rated voltage	24 VDC -15% .. +20%
Voltage range	20 .. 30 V
Load current	max. 0,5 A / output
Potential isolation	none
Protective circuit	Overload, over-voltage, over-temperature
Inductive load protection	Provided
Signal voltage ON	type. V_{supply} - 1,5 V

3.2 Mechanical Dimensions

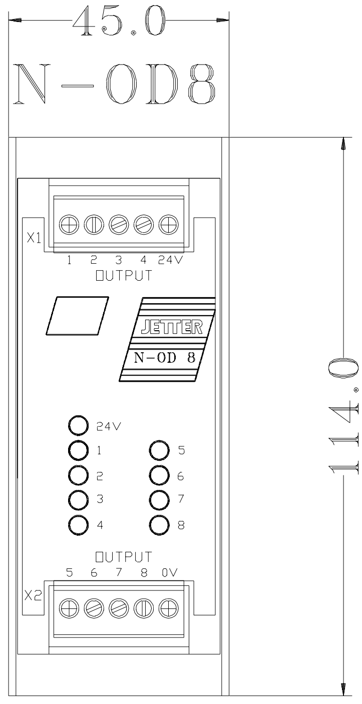


Figure 20: Mechanical Dimensions of the Digital Output Modules

3.3 Description of Connections

Eight output terminal points have been made available for the 24V signal on the expansion module. The 0V signal is connected to the control cabinet ground (GND).

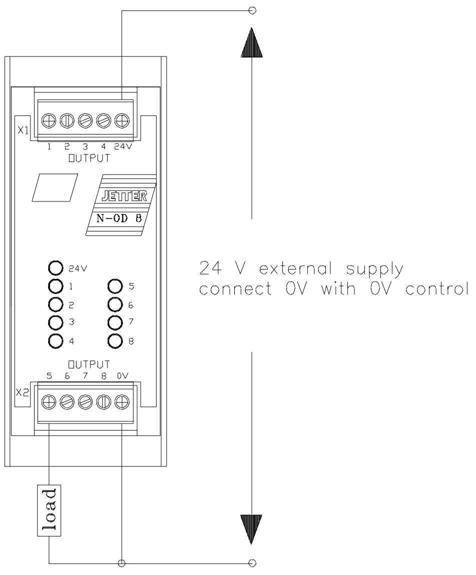


Figure 21: Exemplary Output Wiring of an N-OD8 Module

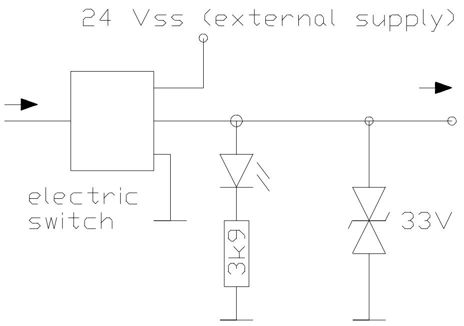


Figure 22: Internal Circuit of the Digital Outputs

Accessing the Digital Outputs

On accessing the digital outputs see *chapter V.1 Accessing the Digital Inputs and Outputs.*

3.4 Description of the LED’s

The LED’s indicate that a 24V output signal has been set on the corresponding output.

4. The N-PS1 Module, Power Supply for Decentralised Modules

4.1 Overview and Technical Data

By the power supply unit, decentralised digital expansion modules are being fed via two terminal points of 24V being changed into 5V logic voltage. Up to five digital expansion modules can be connected to a power supply.

Note:

Only the digital and analogue input and output modules are fed by the N-PS1 power supply, while the intelligent modules have got their own 24V supply.

For the input NANO system bus, a SUB-D male connector, and for the outgoing system bus a 9 pin female SUB-D connector has been provided.

N-PS1: Power Supply for Decentralised Arrangement	
Connection to system bus	Male 9 pin SUB-D connector
24 V connections	Terminal Screws
Voltage supply	24 VDC, -15% - +20%
LED 24 V	24 V attached in the 20 to 30 V range
LED 5V	Internal logical voltage in the 5V ±5%
Dimensions (H x W x D in mm)	114 x 45 x 70
Weight	360g
Mounting	DIN rail

4.2 Mechanical Dimensions

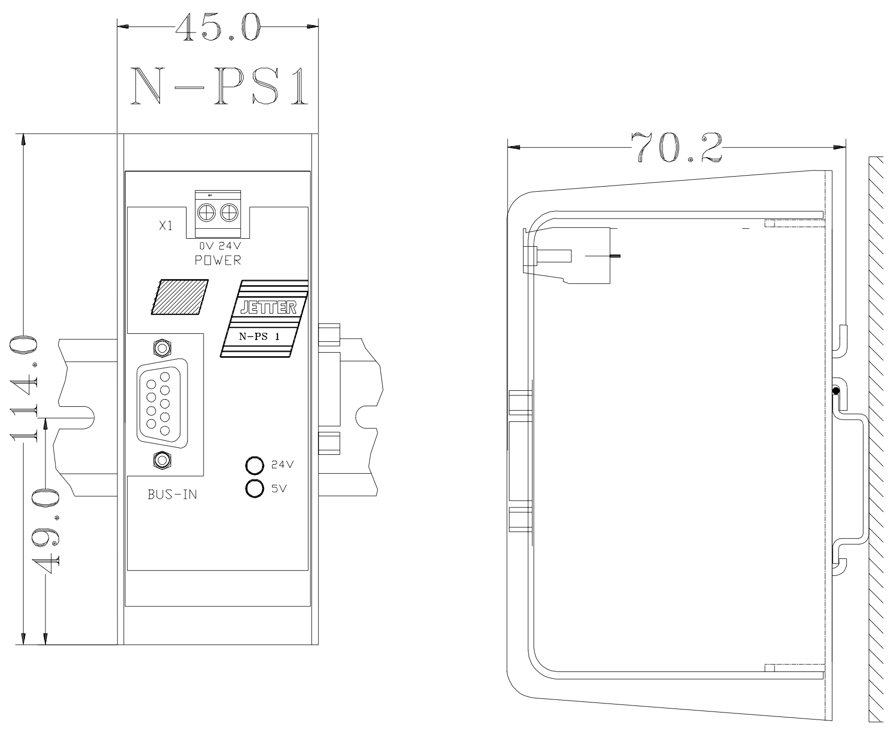
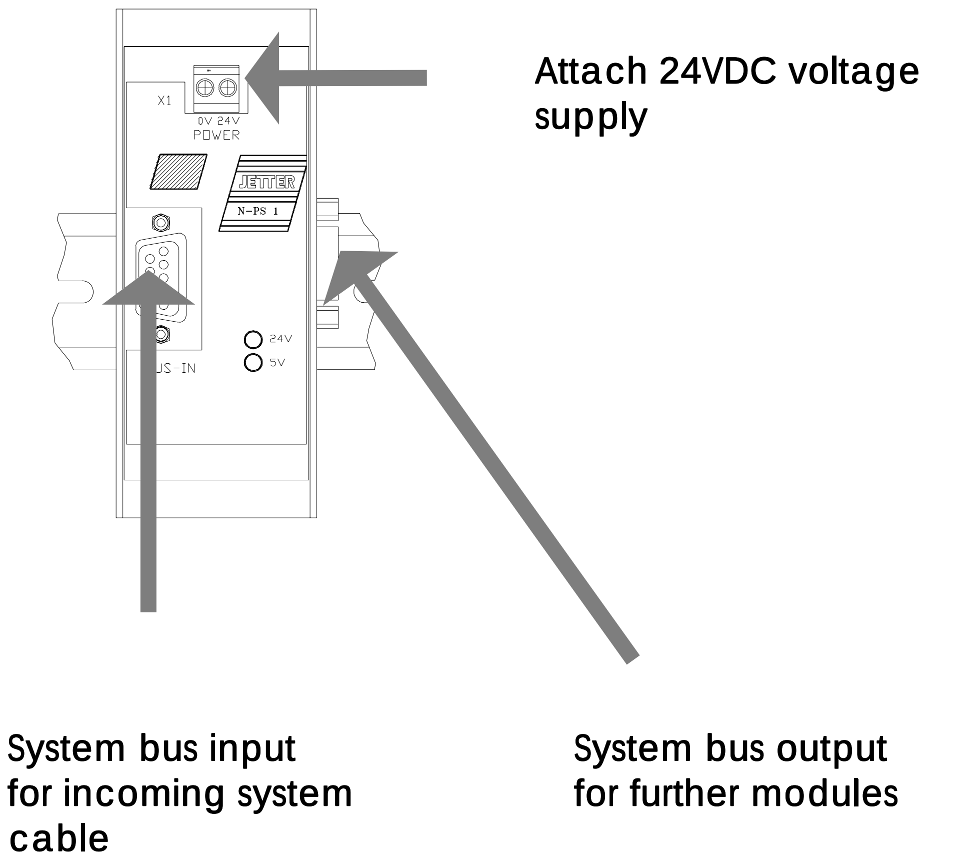


Figure 23: Mechanical Dimensions of the N-PS1 Power Supply Module

4.3 Terminal Description


Attach 24VDC voltage supply
X1
0V 24V
POWER
JETTER
N-PS 1
24V
5V
US-IN
System bus input for incoming system cable
System bus output for further modules

4.4 Description of LED's

LED of the N-PS1 Module	
LED 24V	24V attached in the 20 to 30V range
LED 5V	Internal logic voltage in the 5V ± 5%

V. Programming (Software)

In this chapter, access to registers, inputs, outputs, and flags of the NANO-A will be explained.

Numbering of registers and special registers, flags and special flags, inputs and outputs, will be presented in a table.

Using the following instructions, access to registers, flags, inputs and outputs can be made.

REG_LOAD, REG	Access to registers
FLAG	Access to flags
INPUT	Access to inputs
OUTPUT	Access to outputs

Examples:

```
LOAD_REGISTER              ;register 100 is
[100 with 1234]            ;loaded with 1234
REG 100 =
REG 100 + REG 200          ;the content of
                           ;register 200
                           ;is added to the
                           ;content of register 100
WHEN                       ;when
  FLAG 10                  ;flag 10 is active,
 THEN                      ;the task will be
 ...                       ;continued
IF                         ;if input 101
  IN 101                   ;is active,
 THEN                      ;output 105
   OUT 105                 ;will be set
```

1. Addressing of the Digital Inputs and Outputs

Basic Controller

Numbering of the Inputs on the Basic Controller	
Input	Number
Input 1	101
Input 2	102
...	...
Input 8	108

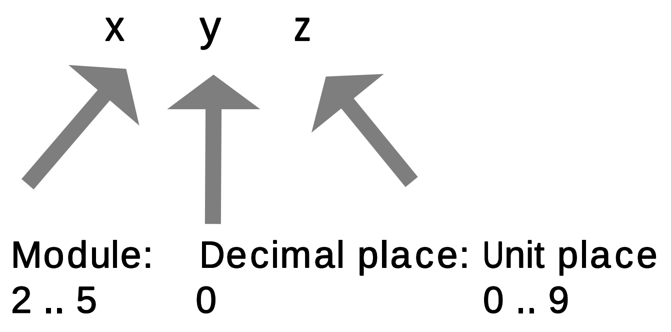
Numbering of the Outputs on the Basic Controller	
Output	Number
Output 1	101
Output 2	102
...	...
Output 6	106

Expansion Modules

The address results from the slot number and the number of the corresponding input / output:

Coding of the Input / Output Number: xyz

Meaning:



Note:

For module numbering, the digital input or output modules are being counted. If there are any analogue input / output modules among the digital ones, they are not being counted.
The basic controller is counted as module number 1; starting from there, the slot numbers are being counted left to right.

Example 1:

Basic controller with two N-ID8 modules and one N-OD8 output module, arranged as follows. The input / output numbering is shown in the table below.

NANO-B Basic Controller	N-OD8 Output Module	N-ID8 Output Module	N-ID8 Input Module
Basic Controller 1	Slot no. 2	Slot no. 3	Slot no. 4
Input 101 .. 108	Output 201 .. 208	Input 301 .. 308	Input 401 .. 408

Example 2:

Basic controller with one analogue expansion module N-AD4, one digital input module N-ID8, and one digital output module N-OD8.

NANO-B Basic Controller	N-OD8 Output Module	N-AD4 anal. Input Module	N-ID8 Input Module
Basic Controller **1**	Slot 2	Slot 3	Slot **4**
Input 101 .. **1**08	Output 201 .. **2**08	Analogue inputs	**! ! !** Input **3**01 .. **3**08

This shows, that for module numbering of the digital inputs and outputs, the analogue module is counted as a void module.

⚠

Note:

For centralised arrangement, first all digital input and output modules are placed in order, and only then the modules for analogue I/Os as well.
For decentralised arrangement, the order is determined by the functional context, which means that intelligent and digital modules can occur together.

2. Access to Flags

2.1 User Flags

Flags 0 to 255 are to the user's free disposal

Flags 0 to 255 are to the user's free disposal. They are also overlapping the registers 2600 to 2610; thus, entire flag groupings can be accessed via registers. In connection with the word-processing instructions W-AND, W-OR, and W-XOR, many new opportunities open up.

Registers	Flags
2600	0 - 23
2601	24 - 47
2602	48 - 71
2603	72 - 95
2604	96 - 119
2605	120 - 143
2606	144 - 167
2607	168 - 191
2608	192 - 215
2609	216 - 239
2610	240 - 255

Note:

Bits 16 to 23 of register 2610 are 0.

Example:

Overlapping of User Flag Registers (Example: Register 2609)									
Bit no.	0	1	2	3	4	...	21	22	23
Reg 2609	1	0	0	0	1	...	0	1	0
Flag	217	218	219	220	221	...	238	239	240

Programming with Flags

Example 1:

Processing is to be started by pressing the start key and the automatic mode being activated by setting the respective flag (for example, in another task).

```
WHEN
    IN iStartKey
    Flag fAutomatic
  THEN
    ...
```

Example 2:

In the main task processing of the second task, which is the automatic task, is to be started using a flag.

```
TASK tMainTask
  ...
    IF
        IN iStartKey
      THEN
        Flag fAutomatic
  ...
        GOTO tMainTask
TASK tAutomatic
```

```
  WHEN
       Flag fAutomatic
    THEN
..
GOTO tAutomatic
```

2.2 Special Flags

In the PROCESS-PLC operating system, various special flags have been provided to support function control or modification. In the table below, an overview over special flags is given, subdivided according to functions, with cross-references to chapters, where the special flags are described in detail in connection with their respective functions.

⚠ Note:

Basically, setting a flag means activating a function. Exceptions will be referred to separately.

Special Flag	Function	Cross-Reference
Control of the User Interface LED's		
2224	LED function key F1	User interface
2225	LED function key F2	manual
2226	LED function key F3	
2227	LED function key F4	
2228	LED function key F5	
2229	LED function key F6	
2230	LED function key F7	
2231	LED function key F8	
2232	LED function key F9	
2233	LED function key F10	
2234	LED function key F11	
2235	LED function key F12	
Querying of User Interface Keys		
2181	SHIFT F1	User interface
2182	SHIFT F2	manual
2183	SHIFT F3	
2184	SHIFT F4	
2185	SHIFT F5	
2186	SHIFT F6	
2187	SHIFT F7	
2188	SHIFT F8	
2189	SHIFT F9	
2190	SHIFT F10	
2191	SHIFT F11	
2192	SHIFT F12	
2193	SHIFT <-	
2194	SHIFT ->	
2198	SHIFT C	
2199	SHIFT ENTER	
2197	SHIFT =	

2223	SHIFT .	
2221	SHIFT -	
2195	SHIFT R	
2196	SHIFT I/O	
2170	SHIFT 0	
2171	SHIFT 1	
2172	SHIFT 2	
2173	SHIFT 3	
2174	SHIFT 4	
2175	SHIFT 5	
2176	SHIFT 6	
2177	SHIFT 7	
2178	SHIFT 8	
2179	SHIFT 9	
2201	F1	
2202	F2	
2203	F3	
2204	F4	
2205	F5	
2206	F6	
2207	F7	
2208	F8	
2209	F9	
2210	F10	
2211	F11	
2212	F12	
2214	<-	
2213	->	
2218	C	
2219	ENTER	
2200	SHIFT	
2217	=	
2222	.	
2220	-	
2215	R	
2216	I/O	
2160	0	
2161	1	

2162	2	
2163	3	
2164	4	
2165	5	
2166	6	
2167	7	
2168	8	
2169	9	
Error Messages Issued by Special Flags		
2048	Timeout I/O module	
2049	Timeout slave module	
2050	Fatal CAN bus error	
Giving Priorities to System Tasks		
2056	PC task after each user task	
2057	LCD task after each user task	
2058	JETWay task	
Network Control by Special Flag		
2064	Shifting between slave registers and registers using 50000-numbers	

3. Register Description (NANO-A Data)

3.1 User Registers

User Registers: 0 to 199

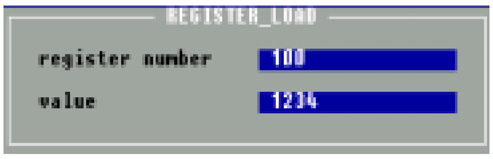
In the range of registers 0 to 199, 200 user registers are to the user's free disposal. They serve as intermediate storage, and as a storage for comparing, measuring, and nominal values.

Power Failure Safe Registers 1000 to 1049. Number of Writing Accesses is Limited

Above that, fifty power failure safe memory registers 1000 to 1049 have been made available. These power failure safe registers are EEPROMS. The number of writing accesses is limited (100000 writing accesses; the number of reading accesses is not limited).

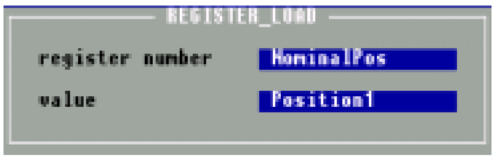
The registers are 24 Bit wide and have a value range of +8.388.607 to -8.388.608.

Registers can, for example, be loaded with the `REGISTER_LOAD` instruction.



```
REGISTER_LOAD
register number   100
value             1234
```

Figure 25: LOAD_REGISTER with Numeric Parameters



```
REGISTER_LOAD
register number   NominalPos
value             Position1
```

Figure 24: LOAD_REGISTER with Symbolic Parameters

Programming with Registers

The instruction

REGISTER_LOAD [x with a]

serves for loading of numeric values (or contents of other registers) into a register.

Description: In the instruction shown above, x represents the number of the register value a is to be written into.

Indirect and Double Indirect Addressing

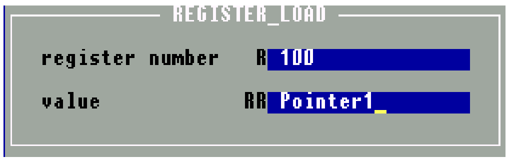
For the "x" and the "a" in the instruction shown above, not only a number can be written, but a register can be specified as well: By pressing the space key an R is written in front of the register number. If "Ry" is written instead of "x", value "a" is written into the register the number of which is in register y.
If "Rb" is written instead of "a", not the value per se, but the contents of the specified register is loaded into register x (or Ry).

If, instead of "a", "RR" (press space key twice) and then any number (b) is written into the register

```
REGISTER_LOAD [ x with RRb]
```

This will have the following result: First, the value of register number b is read. This value then serves as register number. This means, a new value is read in the register having got this value as its number, and finally, this new value is stored in register x.

Indirect and Double Indirect Addressing of Registers


REGISTER_LOAD

register number R 100

value RR Pointer1_

Figure 26: By pressing **(SPACE)** or **(CTRL) (R)** the Indirect Steps R and RR can be Entered.

Examples:

1) Loading of a number into a register

```
REGISTER_LOAD [ rNewPosition with 1280]
```

Value 1280 is loaded into the register `rNewPosition`.

2) Copying one register into another one

```
REGISTER_LOAD [ rVoltage with RrVoltage1]
```

The value written into register `rVoltage1` will be loaded into register `rVoltage`. With other words, register contents `rVoltage1` will be copied into register `rVoltage`.

3 Loading by double indirect addressing

```
REGISTER_LOAD [rVoltage with
RR(rV_Pointer)]
```

The value of register number [contents of register `rV_Pointer`]will be loaded into register `rVoltage`.

Double Indirect Addressing: Example

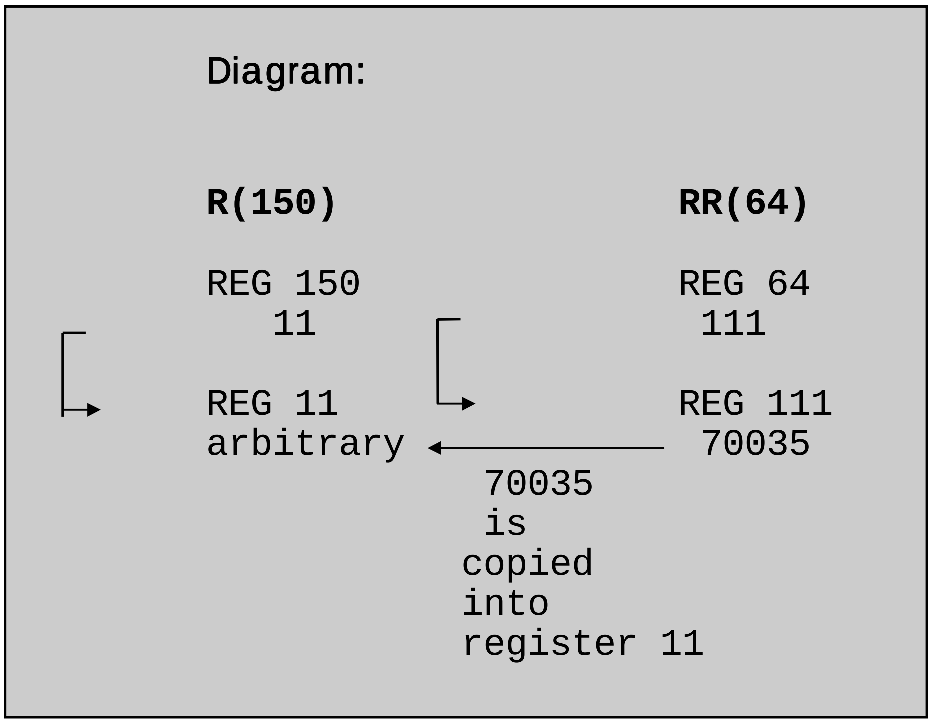
Register Number	Value
REG 64	111
REG 111	70035
REG 150	11
REG 11	arbitrary

with this content, the following instruction will be carried out:

REGISTER_LOAD [R(150) with RR(64)]

The following register values result from this instruction:

Register	64	= 111	(remains)
Register	111	= 70035	(remains)
Register	150	= 11	(remains)
Register	11	= R150	= RR64 = R111 = 70035



Calculating with Registers

The following instructions are used for calculating:

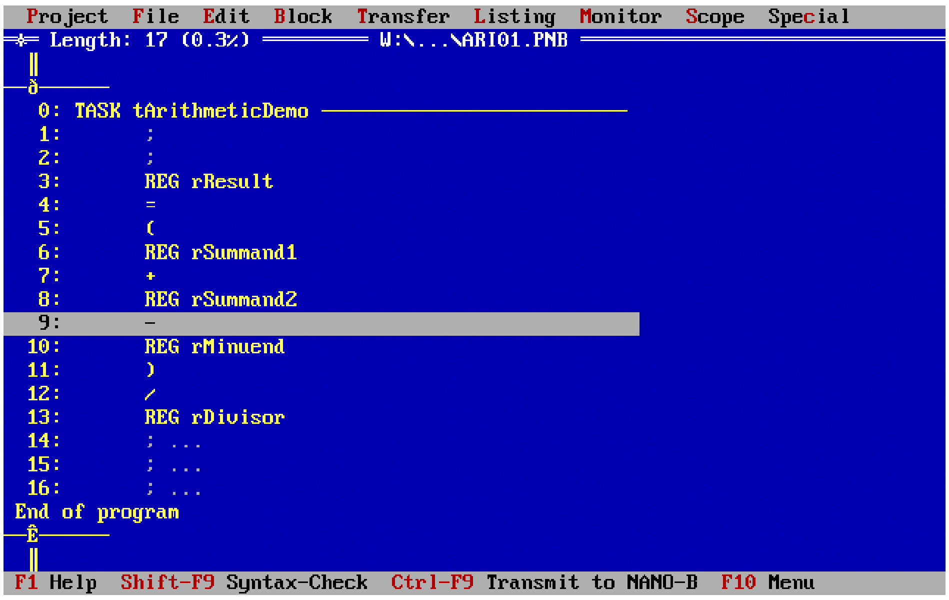
REG <RegNo>

REGZERO <RegNo>

REGDEC <RegNo>

REGINC <RegNo>

In all four instructions it is possible to indirectly specify the register number, which is the only parameter that must be given. As a register number, `R100` can be written. This means that for the parameter the contents of the register numbered `[REG 100]` will be chosen.


Project File Edit Block Transfer Listing Monitor Scope Special
F1 Help Shift-F9 Syntax-Check Ctrl-F9 Transmit to NANO-B F10 Menu

```
Length: 17 (0.3%)          W:\...\ARI01.PNB
TASK tArithmeticDemo
        ;
        ;
        REG rResult
        =
        (
        REG rSummand1
        +
        REG rSummand2
        -
        REG rMinuend
        )
        /
        REG rDivisor
        ; ...
        ; ...
        ; ...
End of program
```

Figure 37: Simple Example on Register Arithmetic

REG

This instruction directly refers to the register value and can be dealt with like a variable. In an output instruction the register left of the equal sign is given a certain value. In an input condition the register content is read. The register accesses on the right of the equal sign in both cases result in reading the register content.

Examples:

1)
```
THEN
   REG 1
   =
   REG 105
   *
   25
```

In this example an instruction (output instruction, introduced by THEN) is shown. Register 105 is read and its contents multiplied by 25. The result of this operation will be stored in register 1. The contents of register 105 will remain unchanged.

2)
```
IF
     REG 1
     =
     REG 105
     *
     25
   THEN
```

In this case the expression `REG 1 = REG 105 * 25` is not part of an output instruction, but of an input condition. In this part of the program the value of register 1 remains unchanged. It will only be compared with the product `REG 105 * 25`.

By using the instruction REGZERO a register is set to 0, or, if queried, whether a register contains a value of 0:

REGZERO <RegNo>

As an input condition, this instruction has got (after IF or WHEN) the following meaning, which is demonstrated in the example below:

Example:

```
IF                  IF
    REGZERO 49      REG 49
  THEN              =
                    0
                    THEN
```

By those two program parts the same functions are carried out. On the right hand side the comparison is to be carried out as a standard arithmetic comparison, and on the left the special instruction REGZERO will be used (advantage: faster execution).

The Instructions

REGDEC   REGINC

Those two instructions serve for decreasing (decrementing) respectively increasing (incrementing) a register by 1. These functions are frequently used in loops for increasing or decreasing counters and pointers.

Examples:

```
1a)                    1b)
      THEN                   THEN
        REGDEC 100           REG 100
                             =
                             REG 100
                             -
                               1
```

Those two program parts have the same function. In both of them the contents of register 100 will be decremented by 1.

```
2a)                      2b)
      THEN                     THEN
        REGINC 88              REG 88
                               =
                               REG 88
                               +
                               1
```

Here the results of both program parts are the same as well. Register 88 is incremented by 1.

```
3)            REGISTER_LOAD [ 1 with 10]
      LABEL   55
              ...
              REGDEC 1
         IF
              REGZERO 1
           THEN
           ELSE
             GOTO 55
           THEN
```

This way a loop can be realised, which is repeated a certain number of times. In the loop the „counting register“ will be decremented by one in each loop, and finally it will be checked whether it is 0 (REGZERO 1). If it is zero, nothing will be executed after the first THEN This means, the program will go to the second THEN to be continued. If register 1 is not zero, though, the program will go back to the starting point of the loop.

3.2 Special Registers

Applied by the Operating System

Special registers are the registers that are used by the operating system for controlling various internal, as well as external, functions.

Special Registers for Messages, User Interfaces, Network, Peripheral Functions

By special registers the functions of the operating system are controlled. Special registers contain (error) reports or serve for controlling user interfaces, peripheral functions, or the instalment of network operation.

Special Registers - as Time Registers, or to Combine Inputs, Outputs, and Flags

There are time registers and special registers, where several inputs, outputs, or flags, have been combined.

⚠

Note:

Improper change of special register settings can lead to malfunctioning or crash.

Overview: Special Registers

Register No..	Function	1) Value Range 2) Reset Value 3) Cross Reference
Operating System (Error) Reports		
2000	Software version	0 .. 65535 Version
2001	Status register	1) -8388608 .. +8388607 2) Status 3) Appendix D.3 Error in the User Program
2002	Runtime register Runtime starting from reset in user time base	1) -8388608 .. +8388607 2) 0
2006	Cycle time of all tasks	1) 0 .. 255 2) not defined
2008	Operating system error	1) 0 .. 65535 2) 0 3) Appendix D2 Operating System: Error Reports
2009	Number of error task	1) 0 .. 255 2) -1 3) Appendix D.3 Error in the User Program
2010	Program address of the error for internal use	1) 0 .. 65535 2) 0
2011	Timeout I/O-module with numbers 2, 3, 4, 5	1) 0 .. 255 2) 0 3) App. D1: Hardware Error
2012	Timeout slave module with module numbers	1) 0 .. 255 2) 0 3) App. D1: Hardware Error
2013	Number of connected I/O modules	1) 0 .. 255 2) Number
2014	Number of connected slave modules	1) 0 .. 255 2) Number
2015	Pointer on module array	1) 0 .. 255 2) 0
2016	Module array, 2015 means pointer 2015 = 0 -> 2016 = number of modules 2015 = 1 -> 2016 = Code of the first module 2015 = 2 -> 2016 = Code of second module, etc. Codes: 0 = N-OD8 1 = N-ID8 128 = N-SV1	1) 0 .. 255 2) Number of modules

	129 = DIMA 255 = not identified	
Task Control		
2004	Task switch conditions	1) 0 .. 255 2) 3 3) Appendix B2 The JETix Mode of Operation
2005	Task timeout time	1) 0 .. 255 2) 20 (20ms) 3) Appendix B2 The JETix Mode of Operation
2007	Number of the highest user task	1) 0 .. 31 2) number
2100 .. 2131	Task status	1) 0 .. 255 2) Status 3) SYMPAS: Index window
2200 .. 2231	Task index	1) 0 .. 65535 2) Beginning TASK 3) SYMPAS: Index window
2300 .. 2331	Task time register	1) -8388608 .. +8388607 2) 0
Various Registers		
2900	Peripheric control registers Bit 1 = 0 Dual channel counter Bit 1 = 1 Single channel counter Bit 0 = 0 No A/D conversion Bit 0 = 1 A/D conversion active	1) 0 .. 65535 2) 0
Control of User Interfaces (LCD Display)		
2804	Number of characters	1) 0 .. 255 2) 48 3) VI. 5. Registers
2805	Number of characters per line	1) 0 .. 255 2) 24 3) VI. 5. Registers
2806	Text choice for DISPLAY_TEXT_2 0 = Text 1, 1 = Text 2	1) 0 .. 255 2) 0 3) VI. 5. Registers
2807	Divisor (USER_INPUT)	1) 0 .. 65535 2) 1 3) VI. 4.
2808	Number of decimal places (USER_INPUT)	1) 0 .. 255 2) 0 3) VI. 4.
2809	Divisor (DISPLAY_REG)	1) 0 .. 65535 2) 1 3) VI. 4.
2810	Number of decimal places (DISPLAY_REG)	1) 0 .. 255 2) 0

		3) VI. 4.
2812	Field width for integer register display	1) 0 .. 255 2) 8 3) VI. 5. Registers
2813	Field width USER_INPUT	1) 0 .. 255 2) 8 3) VI. 5. Registers
2814	Indirect cursor position	1) 0 .. 255 2) 0 3) VI. 5. Registers
2815	Allowed value (Default) USER_INPUT	1) -8388608 .. +8388607 2) 0 3) VI. 5. Registers
2816	Sign suppression	1) 0 .. 255 2) 0 3) VI. 5. Registers
2817	Status of USER_INPUT	1) 0 .. 255 2) Status 3) VI. 5. Registers
2818	Restrictions of monitor functions (0=OFF, 1=ON) Bit0=0 R, I/O keys without monitor function (yet, bits are set) Bit0=1 R key with monitor function Bit1=0 R key without function flag input Bit1=1 R, I/O key with flag input Bit2=0 R, I/O key without output access Bit2=1 R, I/O key with output access Bit3=0 R, I/O key without input access Bit3=1 R, I/O key with input access Bit4=0 = Register contents not changed by key Bit4=1 = Register contents changed by the key Bit5=0 = Flag not changed by the key Bit5=1 = Flag changed by the key Bit6=0 = Outputs not changed by the key Bit6=1 = Outputs changed by the key Bit7=0 = Inputs not displayed by pressing the key Bit7=1 = Inputs displayed by pressing the key	1) 0 .. 255 2) 255 3) VI. 5. Registers
2819	Display time of monitor functions	1) 0 .. 65535 2) 350

		3) VI. 5. Registers
2820	Switch to monitor display	1) 0 .. 255 2) 0 3) VI. 5. Registers
2821	Dialogue language 0=German, 1=English	1) 0 .. 255 2) 0 3) VI. 5. Registers
Network Control		
2700	Network number	1) 0 .. 255 2) 2 3) VI. 5. Register for Network Operation
2701	Baud rate	1) 0 .. 255 2) 10 3) VI. 5. Register for Network Operation
2702	Register offset	1) 0 .. 65535 2) 0 3) VI. 5. Register for Network Operation
2703	Flag offset	1) 0 .. 65535 2) 0 3) VI. 5. Register for Network Operation
2704	Input offset	1) 0 .. 65535 2) 100 3) VI. 5. Register for Network Operation
2705	Output offset	1) 0 .. 65535 2) 100 3) VI. 5. Register for Network Operation
2706	Output mask	1) 0 .. 65535 2) 0 3) VI. 5. Register for Network Operation
Time Register		
2003	Time base for DELAY, START-TIMER, and TIMER-END?	1) 0 .. 255 2) 10 (100ms)
2300 .. 2331	Task time register	1) -8388608 .. +8388607 2) 0
24 combined inputs		
2400	101..108, 201..208, 301..308	
2401	201..208, 301..308, 401..408	
...		
2413	1401..1408, 1501..1508, 1601..1608	
16 combined inputs		

2420	101..108, 201..208	
2421	201..208, 301..308	
...		
2434	1501..1508, 1601..1608	
8 combined inputs		
2440	101..108	
2441	201..208	
...		
2455	1601..1608	
24 combined outputs		
2500	101..108, 201..208, 301..308	
2501	201..208, 301..308, 401..408	
...		
2513	1401..1408, 1501..1508, 1601..1608	
16 combined outputs		
2520	101..108, 201..208	
2521	201..208, 301..308	
...		
2534	1501..1508, 1601..1608	
8 combined outputs		
2540	101..108	
2541	201..208	
...		
2555	1601..1608	
Overlapping register - flag		
0	256 .. 279	
1	280 .. 303	
...		
74	2024 .. 2047	
2600	0..23	
2601	24..47	
...		
2610	240..255	
2611	2048 .. 2071	
2612	2072 .. 2095	
...		
2621	2288 .. 2301	
Single / Dual channel counter		
2901	Counter value	1) -8388608 .. +8388607

		2) 0
2918	Speed	1) -32768 .. +32767 2) 0
2919	Speed Time Base	1) 0 .. 255 2) 0

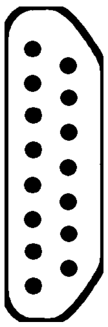
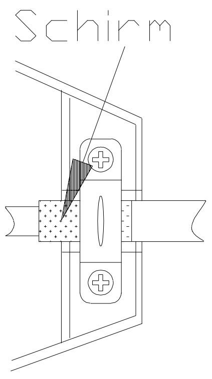
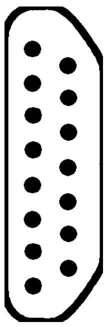
VI. User Interfaces, Operator Guidance

1. Overview, Technical Data

Overview: User Interfaces				
Type	Display	Keys	Remarks	Interface Cable
LCD9a	2 lines of 24 characters each	12 F keys (LED) special function keys decimal block		RS422 DK-422
LCD10a	2 lines of 24 characters each	12 F keys (LED) special function keys decimal block	9mm height of characters illuminated	RS422 DK-422
LCD11	4 lines of 21 characters each	12 F keys (LED) special function keys decimal block	illuminated	OpenColl DK
LCD12	2 lines of 16 characters each	4 F keys special function keys decimal block	designed for operation by manual operation systems	OpenColl DK
LCD16	4 lines of 20 characters each	5 F keys (LED)	can be extended by keyboard modules (NUM25) and handwheel modules (HR1)	RS422 DK-422

Overview: User Interfaces				
Type	Display	Keys	Remarks	Interface Cables
LCD17	Graphic Display 128 x 240 Pixel	6 F keys (LED) special function keys decimal block cursor block	Monitoring with number object text variable bargraph DA-transfer	RS422 DK-422
LCD23 LCD23L LED23	2 lines of 24 characters each 1 line of 16 characters each 1 line of 7 characters	Cursor left Cursor right ENTER	5mm character height 8mm char. height 12mm char. height	RS422 DK-422
LCD25 LCD25L LED25	2 lines of 24 characters each 1 line of 16 characters 1 line of 7 characters (LED)	5 F keys (LED)	5mm char. height illuminated 8mm char. height illuminated 12mm char. height	RS422 DK-422
LCD27	2 lines of 24 characters each	5 F keys cursor block clear ENTER		RS422 DK-422
LCD34	2 lines of 24 characters each	12 F keys (LED) special function keys decimal block	illuminated	RS422 DK-422

2. Terminal Description

User Interface Cable (DK-422)			
PROCESS-PLC			User Interface
15 pin SUB-D male connector 	RS422 max. cable length: 400m  **Please shield extensively! Only use metallised housings!**		15 pin SUB-D-male connector 
Pin	Signal		Pin
4	24 VDC		15
7	Gnd		12
10	SDB	RDB	6
11	SDA	RDA	7
12	RDB	SDB	4
13	RDA	SDA	5

Note:

The pre-fabricated connection cable DK-422, together with the male connector for operating units is supplied by JETTER.
If you make the cables yourself, the following minimum requirements must be met:

Number of wires:	6
Diameter:	$0,25^2$
Male connector:	SUB-D, metallised
Shielding:	as a whole, not in pairs

On both sides of the shield, extensive contact with the male connector housings must be granted.

3. Programming of User Interfaces: DISPLAY_TEXT, DISPLAY_REG, USER_INPUT

In this chapter, the instructions which are necessary for user interface and keyboard modules, will be described.

DISPLAY_TEXT

DISPLAY_REG

USER_INPUT

Display of Texts

The instruction

DISPLAY_TEXT [#<Device no.>, cp=<cursor pos> "<Text>"]

serves for editing texts on user interfaces.

Meaning of the Parameters

Device Number

For this parameter, 0, 1, 2, or 3, can be entered.

#0 or #2

A user interface will be controlled.

#3

This device number causes the controller to edit the text on a printer.
Separate display on several simultaneously connected user interfaces is possible (a description can be ordered).

Cursor Position

By this parameter the cursor position is defined, where the first bit of the text is to appear.

Cursor Positions of Various User Interfaces	
Type	**Cursor Positions**
LCD9	1. line: 1 to 24 2. line: 25 to 48
LCD10	1. line: 1 to 24 2. line: 25 to 48
LCD12	1. line: 1 to 16 2. line: 17 to 32
LCD 16	1. line: 1 to 20 2. line: 21 to 40 3. line: 41 to 60 4. line: 61 to 80
LCD17	status line: 1 to 40
LCD23 LCD23L LED23	1. line: 1 to 24 2. line: 25 to 48 1 to 16 1 to 7
LCD25 LCD25L LED25	1. line: 1 to 24 2. line: 25 to 48 1 to 16 1 to 7
LCD27	1. line: 1 to 24 2. line: 25 to 48
LCD34	1. line: 1 to 24 2. line: 25 to 48

The text will be attached after the last character by cursor position 0

Cursor position 0 has got a special meaning: The latest text is attached to the text edited last. The cursor will appear at exactly the same position where it was last after having carried out the last user interface instruction.

Text

Here, the text can be written which is to be displayed. In this case, the two characters "_" and "$" serve as control characters:

"_" First, deleting the display, and then display of the given text is triggered by this character, starting from cursor position 1 (independent from the parameter that has been input). This character does only make sense, when it appears at the beginning of the text, as otherwise the first part of the text would be displayed first, yet then would be deleted again immediately. This character has got the meaning DELSCR (Delete Screen). If it is to be displayed, the character code for DELSCR can be changed in the special register.

"$" By this character, the rest of a line, following the present cursor position, will be deleted. This character is also called DELEOL (Delete End of Line).

Examples:

1) `DISPLAY_TEXT [#0, cp=0, "_ actual position:"]`

By this instruction, first, the entire LC display is deleted, and after this, "actual position:" is written into the upper line of the user interface (cursor position = 1).

After "cp=", any other number could be written, as it won't be considered after the DELSCR character any more.

```
Actual Position:


```

2) **`DISPLAY_TEXT [#0, cp=25, "nominal position: $"]`**

After issuing this instruction, at the given cursor position 25, i.e. starting with the first character of the second display line, the text "nominal position:" is written; then, the rest of this line is deleted.

3) **`DISPLAY_TEXT [#0, cp=0, " ERROR "]`**

After issuing this instruction, the text "ERROR" is written, starting from the present cursor position. This means that the text is simply attached to the one written last.

The cursor position can be given indirectly using register 2814

If in register 2814 anything else but zero has been written, the register contents is interpreted as cursor position and the text "ERROR" written at this position.

Display of Register Contents

The instruction

DISPLAY_REG [#<Device no.>, cp=<Cursor Pos> Reg=<RegNo>]

serves for the output of a register value on operating devices.

The parameters **device number and cursor position** have got the same function as the DISPLAY_TEXT instruction (see above). Additionally, the **number of the register** the content of which is to be displayed, must be input here. For this purpose, indirect addressing can be applied as well.

Examples:

1) **DISPLAY_REG [#0, cp=17, Reg=100]**

By this instruction, register 100 is transferred to the LC display. If register 2812 has not been changed since reset, register 100 will be displayed at the end of the first display line, as shown below (assumptions: display was empty before the instruction was issued, and register 100 = -3567).

```
.................-    3567
........................
```

The dots are to represent the positions which have still got the previous contents after issuing the instructions.

2) **`DISPLAY_TEXT [#0, cp=25, "actual position :$"]`**
`DISPLAY_REG [#0, cp=41, Reg=11009]`

In this example, useful combination of the two DISPLAY instructions is illustrated: First, the text "actual position:" is written into the second line (left), while the rest of the second line is deleted (Dollar character “$“). By the second instruction, register 1109 is displayed down on the right. In this register, the actual axis position is stored, if a servo controller module has been connected at slot no. 1. This applies to NANO-B only. (Assumptions: The actual position of axis 11 is to have value 5400.)

```
........................
actual position:    5400
```

The dots are to represent the positions, which, after having given the instructions still have got the previous contents.

Query of Register Values by the Program

The instruction

USER_INPUT [#<DeviceNo>,cp=<Cursor pos>, Reg=<RegNo>]

serves for writing the register values which can be input by a user interface.

For the parameters **device number and cursor position** the same facts apply as for the DISPLAY_TEXT instruction, yet with the following alterations: If cursor position 0 is input, the value of register 2814 will be chosen as cursor position at the user input. If this value is 0 as well (which is the reset value of the register), the register contents will be written at the present cursor position.

The **register number** is the number of the register the value that has been input is assigned to. Here, a simple indirect register address is possible as well.

For USER_INPUT there are normally 8 characters available. This value (format of the user input), which has been stored in register 2812, can also be altered.

Example:

```
      DISPLAY_TEXT [#0, cp=1, "_New Position ?"]
      USER_INPUT [#0, cp=17, Reg=100]
```

In order to achieve sensitive operator guidance, the USER_INPUT instruction is most times combined with the DISPLAY_TEXT instruction.

After issuing those two instructions, the NEW POSITION? text will be displayed on the left of the upper line, and after that, the input of a number is being waited for. This number, which will be stored in register 100, will serve as a new nominal position for a positioning run (this applies to NANO-B only).

4. Input and Display of Fixed Point Numbers

Fixed Point Numbers can be displayed and input with the help of the user interfaces.

The functions of registers 2812 (field length for `DISPLAY_REG`) and 2813 (field length for `USER_INPUT`) remains unchanged; thus, the registers are specified as before. If the result is to be displayed, though, the comma is presented as an additional character. This means, that in the display of the result, one more character than it has been defined for the registers is available.

Display of Fixed Point Numbers

For this purpose, two additional special registers have been provided:

Register 2809: Divisor (`DISPLAY_REG`)

By the register value, the number of decimal-places is defined (as an alternative, register 2810 can be used).

2809: Divisor for Value Output DISPLAY_REG	
Register Value	**Post Comma Places**
1	0
10	1
100	2
1000	3
10000	4

4 decimal places are possible as a maximum.

Register 2810: Post Comma Places (DISPLAY_REG)

By the register value, the number of decimal places is defined (as an alternative, register 2810 can be used).

2810: Decimal Places DISPLAY_REG	
Register Value	**Decimal Places**
0	0
1	1
2	2
3	3
4	4

4 decimal places are possible as a maximum.

Only one rgister is to be defined; alternatively, either register 2809 or register 2810 can be written into.

Example:

By the instruction

`DISPLAY_REG [#0, cp=1, reg=200]`

the content of register 200 is displayed on the LCD.

Number **20.00,** for example, is presented by the following register definitions:

Register 200 =	2000	
Register 2809 =	100	(alternative to register 2810)
Register 2810 =	2	(alternative to register 2809)

⚠ Note:

The numeric value of register 200 does not change. Only for presentation on the display a comma is added.

Input of Fixed Point Numbers:

For this purpose, two additional special registers have been made available:

Register 2807: Divisor (USER_INPUT)

In this register, the number of decimal places is stored at the data input (as an alternative, register 2808 can be used).

2807: Divisor for Value Input USER_INPUT	
Register Value	**Decimal Places**
1	0
10	1
100	2
1000	3
10000	4

4 decimal places are possible as a maximum.

Register 2808: Decimal Places (USER_INPUT)

In this register, the number of decimal places is stored at the data input (as an alternative, register 2807 can be used).

2808: Decimal Places USER_INPUT	
Register Value	**Decimal Places**
0	0
1	1
2	2
3	3
4	4

4 decimal places are possible as a maximum.

For evaluation of the number of comma places, either register 2807 or 2808 can be used.

Example:

By the instruction

```
USER_INPUT [#0, cp=1, reg=200]
```

data are written into register 200 by the user interface.

When the operator inputs 20.00, the following values appear in the corresponding registers:

Register 200 =	2000	
Register 2807 =	100	(alternative to register 2808)
Register 2808 =	2	(alternative to register 2807)

Note:

The numeric value of register 200 is 2000. Only on the display of the user interface a comma is shown. (The operator must input the value of register 200, together with the desired comma places. The values of registers 2807 and 2808 result from this input).

USER_INPUT: Suggested Value (Default)

An additional special register has been provided to suggest a value to the user after giving the `USER_INPUT` instruction, which can either be confirmed by ENTER or else be changed.

Register 2815: Suggested Value

The register value will be shown on the user interface with the cursor following, instead with zero. The value can be confirmed by pressing ENTER, or else it can be changed first to be confirmed by ENTER afterwards.
By pressing C (clear), the input is deleted; then the suggested value of register 2815 will appear again.

5. Registers for User Interfaces

Overview: Registers for User Interfaces		
Control of User Interfaces (LCD Display)		
2804	Number of characters	1) 0 .. 255 2) 48 3) VI. 5. Registers for user interfaces
2805	Number of characters per line	1) 0 .. 255 2) 24 3) VI. 5. Registers for user interfaces
2806	Text choice for DISPLAY TEXT_2 0 = Text 1, 1 = Text 2	1) 0 .. 255 2) 0 3) VI. 5. Registers for user interfaces
2807	Divisor (USER_INPUT)	1) 0 .. 255 2) 1 3) VI. 4. Input and display of fixed point numbers
2808	Number of decimal places (USER_INPUT)	1) 0 .. 255 2) 0 3) VI. 4. Input and display of fixed point numbers
2809	Divisor (DISPLAY_REG)	1) 0 .. 255 2) 1 3) VI. 4. Input and display of fixed point numbers
2810	Number of decimal places (DISPLAY_REG)	1) 0 .. 255 2) 0 3) VI. 4. Input and display of fixed point numbers
2812	Field width of integer register display	1) 0 .. 255 2) 8 3) VI. 5. Registers for user interfaces
2813	Field length USER_INPUT	1) 0 .. 255 2) 8 3) VI. 5. Registers for user interfaces
2814	Indirect cursor position	1) 0 .. 255 2) 0 3) VI. 5. Registers for user interfaces

2815	Allowed value (default) for USER_INPUT	1) -8388608 .. +8388607 2) 0 3) VI. 5. Registers for user interfaces
2816	Sign suppression	1) 0 .. 255 2) 0 3) VI. 5. Registers for user interfaces
2817	Status of USER_INPUT	1) 0 .. 255 2) Status 3) VI. 5. Registers for user interfaces
2818	Restriction of monitor functions (0=aus, 1=ein) Bit0=0 R, I/O keys without monitor functions (yet, bits will be set)) Bit0=1 R key with monitor-function Bit1=0 R key without function flag input Bit1=1 R, I/O key with flag input Bit2=0 R, I/O key without output access Bit2=1 R, I/O key with output access Bit3=0 R, I/O key without input access Bit3=1 R, I/O key with input access Bit4=0 = register contents are not changed by key Bit4=1 = register contents are changed by key Bit5=0 = flag is not changed by key Bit5=1 = flag is changed by key Bit6=0 = flag is changed by key Bit6=1 = outputs are changed by key Bit7=0 = inputs are not displayed by key Bit7=1 = inputs are displayed by key	1) 0 .. 255 2) 255 3) VI. 5. Registers for user interfaces
2819	Display time for monitor functions	1) 0 .. 65535 2) 350 3) VI. 5. Registers for user interfaces
2820	Activate monitor display	1) 0 .. 255 2) 0 3) VI. 5. Registers for user interfaces
2821	Dialogue language:	1) 0 .. 255

	0=German, 1=English	2) 0 3) VI. 5. Registers for user interfaces

Register 2804: Number of User Interface Characters	
Function	**Description**
Read	Present value of entire number of user interface characters (all lines) Value after reset: 47
Write	New value for number of characters of the connected user interface
Value Range	1 - 127

The register is initialised by the connected user interface.

Register 2805: Number of Characters per Line	
Function	**Description**
Read	Present value of number of user interface characters Value after reset: 24
Write	New value for number of characters of the connected user interface
Value Range	1 - 127

The register is initialised by the connected user interface.

Register 2806: Text Choice for the DISPLAY_TEXT_2 Instruction	
Function	**Description**
Read	Present value for the text to be output in connection with the `DISPLAY_TEXT_2` instruction. Value 0: Text 1 Value 1: Text 2 Value after reset: 0
Write	New value for text choice: Value 0: Text 1 Value 1: Text 2
Value Range	0 - 1

Bilingual text output is possible

By the `DISPLAY_TEXT_2` instruction, a choice can be made between two texts for the text output. This does, for example, make sense, when the operator guidance is to be bilingual, e.g. text 1 for the customer, text 2 for the service staff. Which of the two texts is displayed will be defined in this register.

Register 2807: Divisor for USER INPUT of Fixed Point Numbers	
Function	**Description**
Read	Present value for the divisor to define the number of decimal places for user inputs: Value 0: no decimal place Value 10: 1 decimal place ...... Value 10000: 4 decimal places Value after reset: 0
Write	-
Value Range	0 - 10000

Decimal Places for the Input of Integer Values

The data being supplied by the NANO-A are integer values. When, at USER_INPUT, they are input with comma places, they can be read by either register 2807 or 2808.
Into register 2807 a divisor has been written which the number of post- comma places result of. A divisor value 10, for example, results in a post comma place (1/10 = 0,1; this relates to a decimal place).

Register 2808: Number of Decimal Places for USER_INPUT of Fixed Point Numbers	
Function	**Description**
Read	Present value of the number of decimal places for user inputs: Value 0: no decimal place Value 1: 1 decimal place ...... Value 4: 4 decimal places Value after Reset: 0
Write	-
Value Range	0 - 4

Display of Decimal Places for USER_INPUT

Different from register 2807, where the number of decimal places is shown by a divisor, in register 2808 the number of decimal places has been written.

Register 2809: Divisor for the Display of Fixed Point Numbers for DISPLAY_REG

Function	Description
Read	Present value for the divisor to define the number of decimal places for DISPLAY_REG: Value 0: no decimal place Value 10: 1 decimal place ...... Value 10000: 4 decimal places Value after reset: 0
Write	New value for defining the number of decimal places for DISPLAY_REG Value 0: no decimal place Value 10: 1 decimal place ...... Value 10000: 4 decimal places
Value Range	0 - 10000

Definition of Decimal Places for Value Output

The data being supplied by the NANO-A are integer values. If these for output on the user interface are displayed by the `DISPLAY_REG` instruction with comma places, this can either be carried out by register 2809 or 2810.
The value of register 2809 is a divisor, which results in the number of decimal places. Divisor value 10, for example, is a decimal place (1/10 = 0,1 refers to a decimal place).

Register 2810: Number of Decimal Places for Display of Fixed Point Numbers in DISPLAY_REG	
Function	**Description**
Read	Present value: Number of decimal places for register display: Value 0: no decimal place Value 1: 1 decimal place ...... Value 4: 4 decimal places Value after reset: 0
Write:	New value: Number of decimal places for register display Value 0: no decimal place Value 10: 1 decimal place ...... Value 10000: 4 decimal places
Value Range:	0 - 4

Direct Definition of Decimal Places for Value Output

Different from register 2809, where decimal places are defined by a divisor, in register 2810 the decimal places to be displayed can be input directly.
If, for example, 3 decimal places are to be displayed, value 3 can directly be input into register 2810. In register 2809, though, the divisor to be input would be 1000.

Register 2812: Field Length for the DISPLAY_REG Instruction	
Function	**Description**
Read	Present field length for the `DISPLAY_REG` instruction. Value after reset: 8
Write	New field length for the `DISPLAY_REG` display
Value Range	0 - 9

Number of Reserved Places for Value Output on the User Interface

Definition of the number of places to be output. Eight places can be reserved for a register display as a maximum.
If only values of two or three characters are to be displayed, only the number of places that are actually needed will be reserved by register 2812.This is of special importance, if a great number of texts and values are to be displayed on a user interface.

Note:

It should be considered that one place each will be occupied by the sign and the comma. If a value of six places is to be output, value 7, respectively 8, is to be input into register 2812.

Register 2813: Field Length for the USER_INPUT Instruction	
Function	**Description**
Read	Present field length for the `USER_INPUT` instruction Value after reset: 0
Write	New field length for the `USER_INPUT` instruction
Value Range	1 - 8

Number of Reserved Places for Value Input

Number of places reserved for input.
If values of only two or three places are to be input, only reservation of the actually needed places on the display will be allowed by register 2813. This is of special importance, if a great number of texts and values is to be displayed on a user interface.

⚠

Note:

It should be considered that one place each will be occupied by the sign and the comma. If a value of six places is to be output, value 7 is to be input into register 2813.

Register 2814: Indirect Cursor Position for DISPLAY_TEXT, DISPLAY_REG and USER_INPUT	
Function	**Description**
Read	Present value for indirect cursor position: Value after reset: 0
Write	New value for indirect cursor position
Value Range	0 - 127

If for the `DISPLAY_TEXT`, `DISPLAY_REG` and `USER_INPUT` instruction the cursor position 0 is input, the cursor position written in register 2814 will be chosen. If the value of this register is 0 as well, the text/value to be displayed will be attached to the texts or values that have been output last.

Register 2815: Allowed Value for the USER_INPUT Instruction (Default)	
Function	**Description**
Read	Present allowed value at the cursor position defined by the `USER_INPUT` instruction: Value after reset: 0
Write	New allowed value for the `USER_INPUT` instruction.
Value Range	-8388608 .. +8388607

When a `USER_INPUT` instruction has been activated, an allowed value will appear at the defined cursor position. As a standard, this value is 0. If at this position another value is to be displayed, it can be written into register 2815.

Register 2816: Sign Suppression at the DISPLAY_REG Instruction

Function	Description
Read	Present value for sign suppression in the case of: Value after reset: 0 (with sign)
Write	Present value for sign suppression: Value 0: with sign Value 1: Sign will not be displayed
Value Range	0 - 1

Register values can be output either with or without sign value. As a standard, output with signs has been activated. Using register 2816, switching to sign suppression is possible.

Register 2817: Status of User Input	
Function	**Description**
Read	Present status of user input Value 0: no user input is active Value 1: user input is active Value after Reset: 0
Write	New status of user input Value 0: interrupt without value transfer Value 2: interrupt with value transfer
Value Range	0 - 2

Interrupt of User Input is Possible

In this register it is shown, whether at the moment a user input is active. Thus, for example, the time of the user input can be monitored by another task. If a defined time is over, an interrupt with transfer of the value shown in the display can be made by writing value 2 into register 2817.
If value 0 has been written into register 2817, an interrupt without value transfer is made.

Register 2818: Keyboard Enable for User Interfaces	
Function	**Description**
Read	Present status of keyboard enable Value after reset: 255
Write	New status of keyboard enable, bitcoded
Value Range	0 - 255

Disable of Keyboard Areas for the User

To definitely enable, respectively disable, the user to have access to the operating functions, certain keyboard areas can be enabled, respectively disabled, by this register.
If keyboard functions disabled for service staff are to be enabled again, this can also be carried out using this register.

Bit of Register 2818	*Function*
Bit 0 = 1	Display of the register by pressing ´R´
Bit 0 = 0	‘R’ key has not got a function
Bit 1 = 1	Display of a flag (2 times ´R´)
Bit 1 = 0	Display of flags is not possible
Bit 2 = 1	Display of outputs by ´I/O´
Bit 2 = 0	Display of flags is not possible
Bit 3 = 1	Display of inputs (2 times ‘I/O´)
Bit 3 = 0	Display of inputs is not possible
Bit 4 = 1	Assignment of values to a register by ´=´
Bit 4 = 0	Assignment of values to a register disabled
Bit 5 = 1	Change of flag condition by ´=´
Bit 5 = 0	Change of flag condition disabled
Bit 6 = 1	Change of output condition by ´=´
Bit 6 = 0	Change of output condition disabled
Bit 7 = 1	Permanent input display by ´=´
Bit 7 = 0	´=´ has got no function for inputs

Register 2819: Shift Time between Monitor Screen and Normal Display

Function	Description
Read	Present value of shift time between monitor screen and normal display: Multiple of 100 ms. Value after reset: 35
Write	New value of shift time between monitor screen and normal display
Value Range	0 - 65536

Shift Time after Application of Monitor Functions

If the monitor functions for register, flag, output and input display, respectively change, have been activated, the display of the user interface will be in the monitor screen mode.
Using register 2819, the time for shifting from monitor screen to normal display can be defined.
Shifting is carried out after having completed the input in the monitor mode of the user interface.
A value of 35 in this register stands for a shift time of 3,5 seconds.

Register 2820: Shifting to Monitor Screen	
Function	**Description**
Read	Present condition: Shift to monitor screen by pressing ´ENTER´ Value 0: Shifting by ´ENTER´ Value 1: Shifting by ´ENTER´ disabled Value after reset: 0
Write	New status of shifting to monitor screen by pressing ´ENTER´ Value 0: Shifting by ´ENTER´ Value 1: Shifting by ´ENTER´ disabled
Value Range	0 - 1

By pressing the ´ENTER´ key, direct shift to monitor screen can be carried out. This function can be activated or deactivated using register 2820.

Register 2821: Display Language	
Function	**Description**
Read	Present setting for the language of integrated user interface functions: Value 0: German Value 1: English Value after Reset: 0
Write	New setting of the language for communication with the user interface: Value 0: German Value 1: English
Value Range	0 - 1

Setting the language for functions of communication between user interface and user. These are the operating system functions of the user interface, though not the text output by the user. Operating system functions of the user interface are, for example, the monitor functions for registers, flags, inputs and outputs.

6. Flags being used by User Interfaces

Flags 2057: LCD Operation after each User Task	
Function	**Description**
Read	Present priority of the user interface: Flag = 0 : User interface will be operated after execution of all user tasks, low priority Flag = 1: User interface will be operated after execution of each individual user task, high priority Value after reset: 0
Write	Setting for high priority, deletion for low priority of the user interface

Definition of the User Interface Priority

The user interface is operated by a kind of background task. In most situations, the user interface has got a priority lower than the priority of the user program. In this case, the user interfaces will not be processed before complete processing of all user tasks. As a rule, this is completely sufficient, because this will happen in a range of centiseconds, which will not be felt to be a waiting time by the user.
If a great number of values are being displayed, though, especially on displays of four lines, while at the same time user inputs are being waited for, the user interface priority can be increased by setting this flag. In this case, the user interface will be operated after each user task. Processing of the operating system will be carried out as follows: Task1, user interface, task 2, user interface, task 3, user interface, etc. For further details

on task processing, please see the register descriptions for task control.

Note:

Normally, the user interface of low priority should be operated.
If during user input there are remarkable delays, the priority of the user interface can be increased by setting the flag. In most cases, more complex user and display functions in the manual or setting mode of the device are needed. Thus, it is possible to set this flag in the manual mode (high priority), and to delete it again in the automatic mode (low priority).

Controlling the Keys and LED's of the User Interface

Controlling the User Interface LED's

LED, Key	Special Flag	LED, Key	Special Flag
LED F1	2224	LED F7	2230
LED F2	2225	LED F8	2231
LED F3	2226	LED F9	2232
LED F4	2227	LED F10	2233
LED F5	2228	LED F11	2234
LED F6	2229	LED F12	2235

Query of User Interface Keys

Key	Special Flag	SHIFT + Key	Special-Flag
Function Keys			
F1	2201	SHIFT-F1	2181
F2	2202	SHIFT-F2	2182
F3	2203	SHIFT-F3	2183
F4	2204	SHIFT-F4	2184
F5	2205	SHIFT-F5	2185
F6	2206	SHIFT-F6	2186
F7	2207	SHIFT-F7	2187
F8	2208	SHIFT-F8	2188
F9	2209	SHIFT-F9	2189
F10	2210	SHIFT-F10	2190
F11	2211	SHIFT-F11	2191
F12	2212	SHIFT-F12	2192

Keys for Special Functions			
<-	2214	SHIFT <-	2193
->	2213	SHIFT ->	2194
C	2218	SHIFT C	2198
ENTER	2219	SHIFT ENTER	2199
SHIFT	2200		
=	2217	SHIFT =	2197
.	2222	SHIFT .	2223
-	2220	SHIFT -	2221
R	2215	SHIFT R	2195
I/O	2216	SHIFT I/O	2196
Numeric Keys			
0	2160	SHIFT 0	2170
1	2161	SHIFT 1	2171
2	2162	SHIFT 2	2172
3	2163	SHIFT 3	2173
4	2164	SHIFT 4	2174
5	2165	SHIFT 5	2175
6	2166	SHIFT 6	2176
7	2167	SHIFT 7	2177
8	2168	SHIFT 8	2178
9	2169	SHIFT 9	2179

VII. Network Operation

1. JETWay-R: Processing Level

The JETWay-R network has two functions:

1. The hierarchic networking of PROCESS-PLC control systems
2. The connection of decentralised peripheral devices like Remote I/Os or valve blocks.

The maximum number of participants per level is 126. This is a monomaster network, which means that on each hierarchic level there is one master and up to 126 slaves.

Note:

The PROCESS-PLC NANO-A is always a slave in the JETWay-R, because it cannot actively access another controller. Nevertheless, other controllers can access the NANO-A.

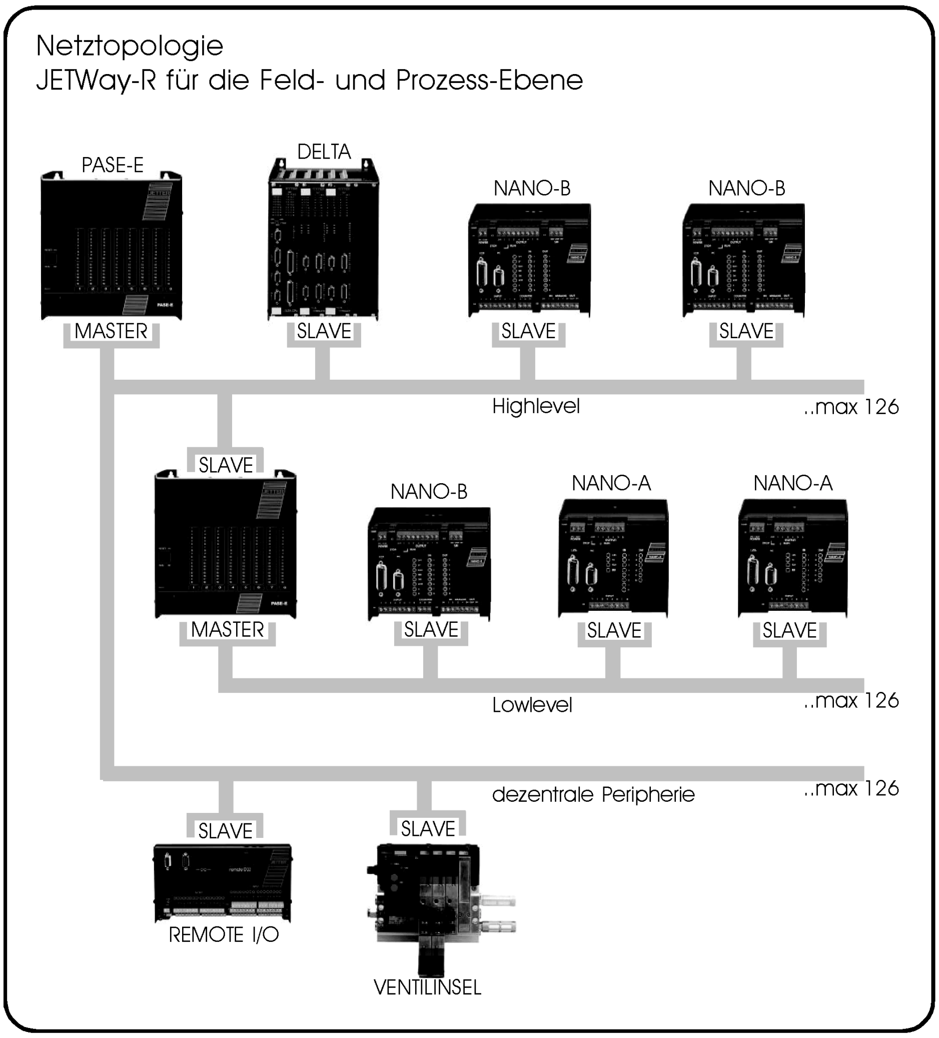
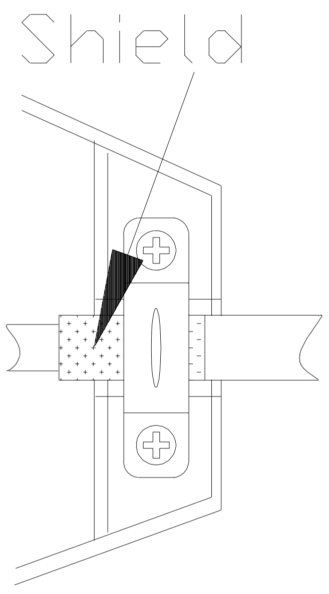


Figure 27: JETWay-R for the Process Level

2. Description of Connections

JETWay-R serves for the networking of several PROCESS-PLC's and/or the networking of Remote I/Os, valve blocks, etc. with a PROCESS-PLC.

JETWay-R Cable		
Connection on the Side of the NANO-B	Shielding	Specification, max. Length
9 pin SUB-D male connector or 15 pin SUB-D male connector	 **Please shield extensively! Only use metallised housings!**	RS485 max. cable length: 400m
Pin	Signal	Remarks
7	Gnd	
8	Data +	
9	Data -	

Note:

For manufacturing this cable the minimum requirements are:

Number of wires:	3
Diameter	$0,25^2$
Male connector:	SUB-D, metallised
Shielding:	as a whole, not in pairs

On both sides, the shield must be extensively connected to the male connector housings.

3. Network Access by 50 000-er Numbers

Note:

The PROCESS-PLC NANO-A is always a slave in the JETWay-R; it cannot actively access another controller. Nevertheless, the NANO-A can be accessed by other controllers.

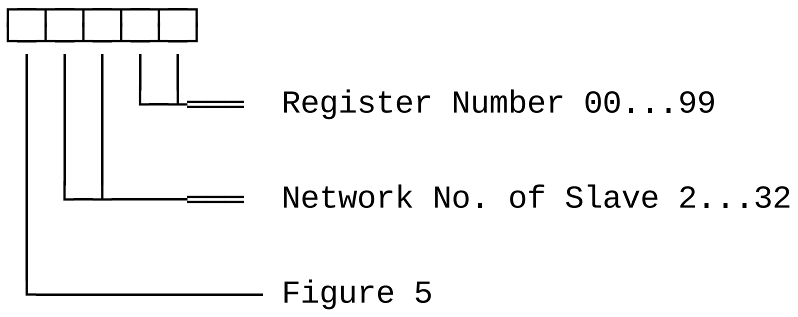
In the following examples the NANO-A is always the slave controller.

3.1 Addressing the Registers

Addressing registers of a controller by the master control only differs from an internal REGISTER_LOAD instruction by the parameter number. Apart from this number, the program sequences for addressing an internal register and for addressing a slave register, are the same.

Register Number 00...99 (for access to NANO-A)

The register number is made up according to the following pattern:



The register offset register is in the NANO-A slave control

Register 2702:

Number offset for registers - NANO-A; the register is on the slave control (NANO-A).

This value is added to the register number in the master control program. The value of the total results is the register number in the NANO-A slave control, which is actually accessed by the master control.

Using these register numbers the master control can address all registers of the slave control.

Example:

Register 62 of the slave control with network number 32 is addressed from a PROCESS-PLC by the instruction

```
REGISTER_LOAD [ 100 with R(53262) ]
```

If a register is to be addressed the number of which is greater than 99, a numeric offset value is to be written into register 2702 of the slave control. This value will be added to the register number in the program of the master control, when registers of the slave control are addressed by the master control.

The instruction

```
REGISTER_LOAD [ 100 with R(53262) ]
```

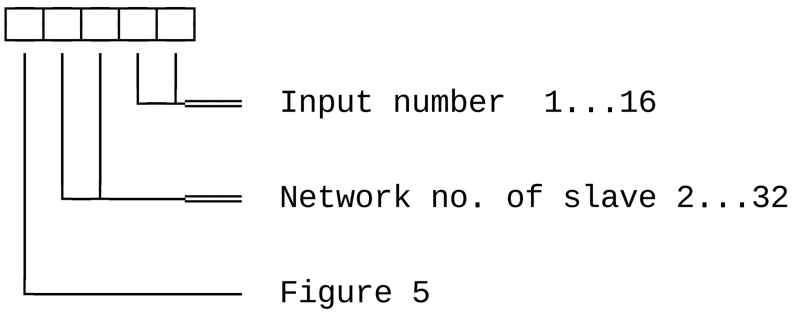
in the master control program, plus value 100 written into register 2702 of the slave control of network number 32 serves for actually addressing register 162 of the slave control.

3.2 Addressing of Inputs, Outputs, and Flags

Addressing of Inputs

Access to inputs of the slave control by the master control only differs from an internal master input instruction by the parameter number. Apart from this number, the program sequences for access to a master input and a slave input are identical.

The input number is made up as follows:



The input offset register is in the NANO-A slave control

Register 2704:

Number offset for inputs - NANO-A; the register is on the slave control (NANO-A).

This value is added to the input number in the master control program. The value of the total results is the input number in the NANO-A slave control, which is actually accessed by the master control.

Value after Reset: 100

Example:

Input 112 in the NANO-A slave control with network number 4 is addressed by the master control by

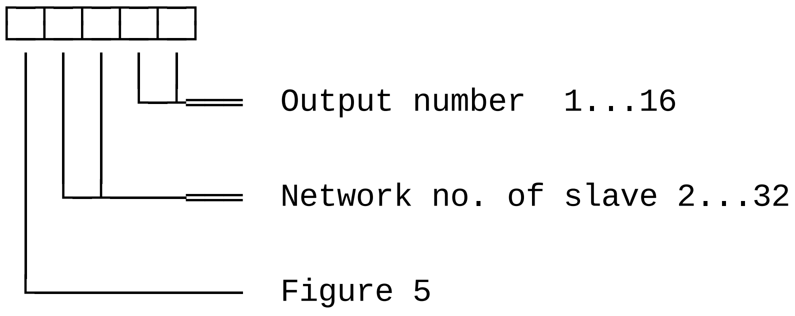
```
INPUT 50412.
```

Before that, value 100 must be written in the number offset register for input no. 2704 (on the NANO-A slave control).

Addressing the Outputs

Access to outputs of the slave control by the master control only differs from an internal master output instruction by the parameter number. Apart from this number, the program sequences for access to a master output and a slave output are identical.

The output number is made up as follows:



The output offset register is in the NANO-A slave control

Register 2705:

Number offset for output NANO-A; the register is on the NANO-A slave control.

This value is added to the output number in the master control program. The value of the total results is the output number in the slave control, which is actually accessed by the master control.

Value after Reset: 100

Example:

Output 113 in the NANO-A slave control with network number 5 is addressed by the master control by

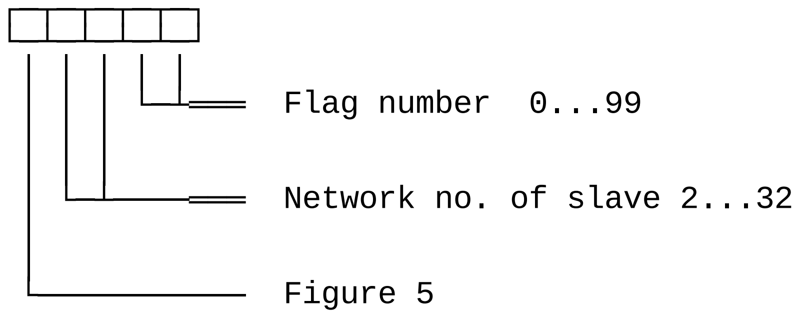
```
OUTPUT 50513.
```

Before that, value 100 must be written in the number offset register for input no. 2705 (on the NANO-A slave control).

Addressing the Flags

Access to flags of the slave control by the master control only differs from an internal master flag instruction by the parameter number. Apart from this number, the program sequences for access to a master flag and a slave flag are identical.

The flag number is made up as follows:



The flag offset register is in the slave control

Register 2703:

Number offset for the NANO-A flag; the register is on the slave control (NANO-A).

This value is added to the flag number in the master control program. The value of the total results in the flag number in the slave control, which is accessed by the master control.
Value after Reset: 0

Example:

Flag 154 in the NANO-A slave control with network number 12 is accessed by the master control by

```
FLAG 51254.
```

Before that, value 100 must be written into the number offset register flags (on the slave control).

4. Network Access by **N-SEND REGISTER** and **N-GET REGISTER**

Note:

These register numbers are not influenced by the number offset in register 2702.

Note:

The NANO-A PROCESS-PLC is always a slave in the JETWay-R; it cannot actively access another controller. Nevertheless, the NANO-A can be accessed by other controllers.

In the following examples, the NANO-A is always the slave controller.

The N-SEND REGISTER Instruction

By the following instruction, registers can be written into slave controls by the master control:

`N-SEND REGISTER [to <PASE no.> from reg<source reg> into reg<destination reg>]`

PASE no.

PASE no. stands for the network number of the slave control which is to be addressed via the network.

Source reg

Here, the number of the register is assigned. This is the register the value of which is to be transmitted to a slave by the network.

Destination reg

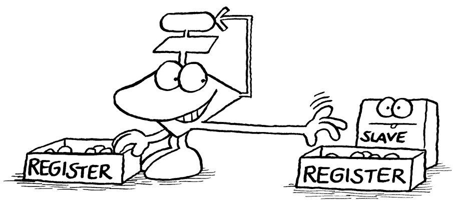
Here, the number of the register is assigned which the contents from the master control is transferred into. This register is on the slave control of the slave number PASE-no.

Example:

```
    N-SEND REGISTER [To 2 from reg100 into reg200]
```

Result: The value of master control register 100 will, after this instruction, be written in slave control register 200 of network number "2".

The N-GET REGISTER Instruction



By the following instruction slave control registers can be read by the master control:

```
N-GET REGISTER [From <PASE no.> reg<source reg>,
                          into reg here =
                          <destination reg>]
```

PASE NO.

In PASE NO. the network number of the slave control is written, which is to be addressed via network.

Source Reg

From source reg the number of the of the register can be read, from which a value is to be written into the master control. This register is in the slave control.

Destination Reg

From destination reg the number of the master control register can be read the value of the slave register is to be written into.

Example:

```
N-GET REGISTER [from 2 Reg200, Reg. here=100]
```

Result: Value of slave control register 200 (network number 2) is copied into master control register 100 by this instruction.

5. Registers for Network Operation

Overview: Network Registers	
2700	Network number
2701	Baud rate
2702	Register offset
2703	Flag offset
2704	Input offset
2705	Output offset

Each PROCESS-PLC system has at least one interface for networking via the JETWay network. The registers described here serve for definition of transfer parameters and participant numbers of this RS485 interface.

Register 2700: Participant Number	
Function	**Description**
Read	Present participant number in JETWay Value after reset: 0
Write	New participant number in JETWay: Value 0: deactivated Value 1: prohibited, as it is a master number Value 2 - 127: possible slave number
Value Range	0 and 2 - 127

The NANO-A can only function as slave in a JETWay-R network. Thus, this participant number must be a slave number. Slave numbers range from 2 to 127.

Register 2701: Baud Rate JETWay-R

Function	Description
Read	Present value for baud rate in JETWay-R Value after reset: 10 (115,2 kBaud)
Write	New value for baud rate in JETWay-R 0 = 150, 2 = 300, 3 = 600, 4 = 1200, 5 = 2400, 6 = 4800, 7 = 19200, 8 = 38400, 9 = 57600, 10 = 115200
Value Range	1 - 65536

Register 2702: Register Offset

Function	Description
Read	Present value for the register offset Value after reset: 0
Write	New value for the register offset
Value Range	0 - 65535

This value will be added to a 50 000-number network access. See *Chapter VII.3.1 : Addressing the Registers.*

Register 2703: Flag Offset	
Function	**Description**
Read	Present value for flag offset Value after reset: 0
Write	New value for flag offset
Value Range	0 - 65535

This value will be added to the flag number of a 50 000-er number network access. *Chapter VII. 3.2 : Addressing Inputs, Outputs, and Flags*.

Register 2704: Input Offset	
Function	**Description**
Read	Present value for input offset Value after reset 100
Write	New value for input offset
Value Range	0 - 65535

This value will be added to the input number of a 50000-number network access. *Chapter VII. 3.2 : Addressing Inputs, Outputs, and Flags.*

Register 2705: Output Offset	
Function	**Description**
Read	Present Value for Output Offset Value after reset: 0

Write	New Value for Output Offset
Value Range	0 - 65535

This value will be added to the input number of a 50000-number network access. See *Chapter VII 3.2 : Addressing Inputs, Outputs, and Flags.*

VIII. Single Channel Counter

1. Description of Connections

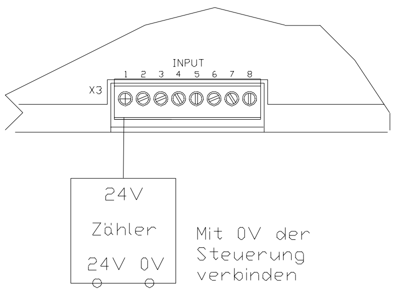
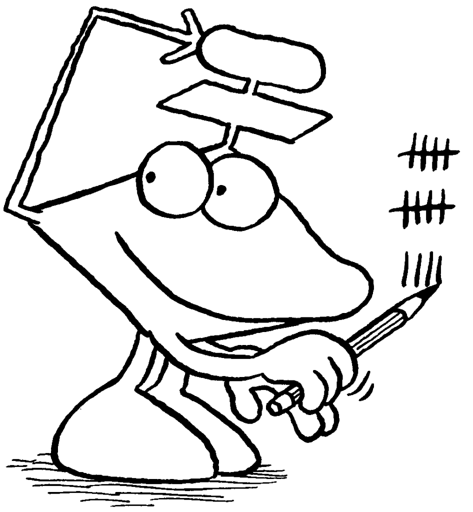


Figure 28: Connection of the single channel counter

The digital input INPUT 1 is connected.



2. Register Description

Register 2900: Peripheral Control Register	
Function	**Description**
Read	Present value of peripheral control register Value after reset: 0
Write	New value of peripheral control register
Value Range	0 - 65535
Function	
Bit 1 = 0 Bit 1 = 1 Bit 0 = 0 Bit 0 = 1	Single channel counter OFF -> dig. input Single channel counter ON A/D conversion deactivated A/D conversion active

The digital input INPUT 1 serves acquisition of events that have a frequency of up to 10kHz. If the single channel counter has been deactivated, only INPUT 1 will function as a digital input.

Register 2901: Counter Value Single Channel Counter	
Function	**Description**
Read	Present counter value Value after reset: 0
Write	Counter value is overwritten
Value Range	24 Bit

Appendix

Further information on, or deepening of, certain topics

This appendix serves for the deepening of certain topics, or for giving an overview over peripheral topics. On some of the topics mentioned here, individual manuals or brochures have been provided. They will explicitly be mentioned in the respective passages.

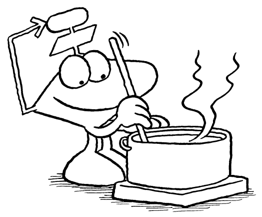
In appendix E you will find a questionnaire about this manual, which can be copied and faxed. We would be very grateful, if you could help us with suggestions and criticism, in order to make the manuals still more user-friendly.

Appendix A: Operating System Update

JETTER has supplied operating system updates (*.BIN or *.HEX files) on the mailbox. These can be inscribed on the respective EPROM and plugged into the controller.

Mailbox: 00 49 / 7141 / 59834

Appendix B: The NANO-A Multitasking Operating System



This chapter has been written for the user who wants to know more about the basic functioning of the NANO-A multitasking operating system.

B1: Basics on Multitasking

Most control systems make use of a program, which is run through in cycles. Cyclic storage run is necessary, unless several parallel programs are applied, which is called multitasking.

Every control system, however small it may be, contains parallel functions and processes. Even, if only one automatic run is needed, there are parallel functions or operator guidance functions to be monitored.

Parallel Functions Carried out by Multitasking

The most practicable way to operate parallel processing is multitasking. The reasons, why this kind of technology has not been applied on a wider range yet, are the following:

1. PLC automation technique is very much obliged to its traditional concept, part of which are the PLC languages ladder diagram, block diagram, and statement list.
2. The standard realtime multitasking operating systems are very complex, so that effective, and thus expensive, hardware will be needed. Besides that, specialists are needed for maintenance.
3. The multitasking operating systems generally known in offices are in realtime only to a certain limit, as many system functions, such as hard disk access, mouse

handling, and similar functions, work as interrupts of a program flow.

4. Due to the complexity of the current multitasking operating systems, application in the area of small to middle-sized control systems has not been possible so far.

Yet, multitasking as such is the way of parallel processing that is easiest to understand and to realise logically.

Display of the Actual Process

When multitasking is applied, transfer into a cyclic program run, which does not correspond to the actual process, is omitted. By multitasking, a way of controlling is granted that refers to the actual process.

JETix: The Multitasking Operating System for Automation Technique

To create an operating system with multitasking and a descriptive, process oriented execution for the whole range of automation technique, JETTER has developed a multitasking operating system that is easy to manage: JETix.

This operating system designed for the requirements of automation technique even allows implementation in the NANO-A mini-size controller.

B2: The Way JETix Works

Multitasking with Single Processor System

First of all a distinction must be made. In the area of electronic data processing, there are so-called multi processor systems (transputer) for applications with a great amount of data, e.g. complex data processing. There, parallel processing is carried out by several processors.
Yet, these are not applied for the multitasking systems generally used in offices, as well in most of the other areas of data processing. Multiprocessor systems are not practicable for the broad spectre of controller applications, due to their hardware, software, and thus financial expenses. Some special applications are an exception. For this reason, a processor to manage all parallel programs is used in automation technique. One of these is JETix.

Some basic techniques are applied to multitasking operating systems. One of them is the so-called time slice or time-sharing technique.

JETix functions by an optimised time-slice technique

In time-sharing, some definite amount of time is made available to each task, which will be processed, until this given time has expired. Then, there will be a change to the next task. Tasks are being changed, until it is the turn of task no. 1 again to start the course from the beginning.

The multitasking that is applied to PROCESS-PLC systems is an optimised or condition dependent time-sharing technique. Up to 32 tasks (parallel programs) can be written. In most cases, especially for mini-controllers, 3 to 10 tasks are realistic.

Note:

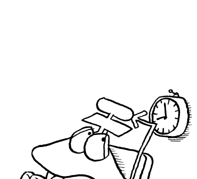
A program <u>must</u> be started with **`TASK 0`**. This is the only task that has to be there from the beginning. The order, in which the further tasks are programmed is of no importance. For reasons of clarity, a complete sequence corresponding to the process should be recommended.

Remark:

As the program processing time does in the first place not depend on the program length, but on the number of tasks that have been used, as few tasks as possible should be implied.

There are fixed and freely definable task switch conditions

The available time is not necessarily made use of by a task. If, for example, the next task instruction to be carried out is an expired DELAY, task switch takes place immediately. After the following instructions, task switch is inevitably carried out (this step cannot be influenced):



- `DELAY` process has not been completed yet
- `WHEN` condition has not been fulfilled yet
- `USER_INPUT` wait, until value is input by the user

Additionally, further task switch conditions can be defined in register 2004, which are:

- when the time of register 2005 has expired
- when the task reaches a `GOTO` instruction
- when the task reaches instruction `IF,` while the condition has not been fulfilled yet

In addition to the user tasks, three further functions are carried out in the background:

- Interface for connection with the user interface
- Interface for connection with VIADUKT or graphic user interface
- JETWay interface

Priority of managing the user interface and the serial interface can be defined

In standard setting, both functions are executed after having carried out all tasks. By flags 2056 and 2057, priority of these functions can be increased. In this case, the interfaces are always addressed in between two task changes.
In most cases, standard setting is best, as the highest priority is usually given to automatic functions instead of operating functions. It is advisable to switch these flags, for example, from automatic function to manual operation of the system.

Remarks:

By the DELAY 0 (Parameter 0) instruction, task switch is enforced. If a task hits DELAY 0 during its processing, it will immediately go to the next task.

Using this instruction, tasks of low priority or program parts can be defined as such. If one or several DELAY 0 instructions are written into the user interface task, the time gained here is made available to the other tasks.

Appendix C: Symbolic Programming

All parameters can be programmed either in numeric or in symbolic mode:

Without Symbols

```
...
THEN
  OUT 302
...
```

With Symbols

```
...
THEN
  OUT oEject
...
```

Due to the use of symbols, the - already good - readability of PROCESS-PLC programs has been further improved.

The symbolic way of writing can be recommend-ed, because it is easier to understand

A fixed way of writing is quite helpful. In the example given above, the name of the output begins with oXxxx. This is a free definition. Basically, any way of writing would be possible, yet, we would recommend a uniform way of writing the beginning of a name, as this is to make clear, whether it is the name of a flag, an input, an output, or a register.
The advantage of this uniform way of writing is the use of names that are known and clear to everybody. If, for example, in a company, several members of staff do the programming, each one will know which kind of parameter is being dealt with. This is also helpful for programs that have been taken over from other companies.

Remarks:

Further information can be taken from the SYMPAS programming manual. Suggestions for the naming of various data and parameters are also made there.

We would recommend to apply this suggested way of writing, as it is also used by our hotline staff.

Appendix D: Bugfix

When dealing with bugfix, first of all the bug must be classified:

- hardware bug
- error reports of the operating system
- bugs in the user's program

D1: Hardware Bugs

If communication with a module connected to a NANO-B is not possible, this will be signalised:

1. In register 2011, respectively 2012, the number of the module will be written, where a communication timeout has occurred.

2. The ERR-LED on the NANO-A basic controller will shine.

D2: Error Reports of the Operating System

The following operating system errors are signalised in register 2008:

Register 2008 Error Description	
Bit Number	Error Description
2	there is no program stored in the NANO-A memory
3	I/O module timeout
5	illegal opcode in the EEPROM
6	wrongly programmed arithmetic calculation
7	a label number has been used several times
8	general syntax error
9	one or more than one output drives are overloaded

D 3: Bugs in the User Program

The best thing is to let the syntax check in SYMPAS remain active, as thus the greatest bugs will be sorted out at once.

In the SYMPAS programming surface, a syntax check to deal with bugs in the user program is implied. This syntax check can be freely activated or deactivated for program transfer to the controller.
If the syntax check has been deactivated, faulty programs can be transferred to the NANO-B. In this case, these errors will be reported in register 2008. In register 2001 it is signalised, whether the program is run normally, or whether it has been stopped.

Register 2001: Status Register	
Function	Description
Read	Condition: Value 0: Program has been stopped Value 1: 0 = 1: Program is running
Write	Value 0: Stop the program Value 1: Start the program
Value Range	0 - 1

In the status register it is signalised, whether the program in the controller is running or stopped at the moment. A program can be stopped, when a syntax error has been detected in the user program. The error type will be reported in register 2008 and the ERR LED will shine. A program can also be stopped via the SYMPAS setup monitor, or by writing into this register.

⚠

Note:

In the LED RUN it is signalised, whether a program is running normally, or whether it has been stopped.

LED RUN activated:	Program is running
LED RUN deactivated:	Program has been stopped

Register 2009: Bug Task Number

Function	Description
Read	Task number, where a bug has occurred.
Write	not permitted
Value Range	0 - 31

If in the user program a bug has been detected, the number of the task, where the bug has occurred, can be read here.

D 4: The JETTER Hotline

If there are any problems which cannot be solved with the help of the manual, our free-of-charge hotline is available for you:

Applications:	**0049 - 7141 / 2550 - 444**
Technical Sales:	**0049 - 7141 / 2550 - 433**
E-mail:	**jetter@jetter.de**

Appendix E: Questionnaire on the NANO-A Manual

Would you, please, evaluate our manual according to the following criteria and fax this questionnaire to us.

Fax number: 0049-7141 / 2550-425

Questionnaire on the NANO-A manual: Please fill in and fax:

0049-7141 / 2550-425

Graphic Layout

o good o quite useful o only medium o bad quality

Commentary:

Structuring

o good o quite useful o only medium o bad quality

Commentary:

Clarity

o good o quite useful o only medium o bad quality

Commentary:

Completeness

o good o quite useful o only medium o bad quality

Commentary:

Index

o good o quite useful o only medium o bad quality

Commentary:

Sequence of Topics

o good o quite useful o only medium o bad quality

Commentary:

Appendix F: Further JETTER Components, Service

Besides controllers, a whole range of peripheral products, or various kinds of service, is also offered by the JETTER GmbH.

About all the components mentioned below, there are brochures available, which can be ordered.

F1: User Interfaces and Monitoring Devices

In addition to our controllers, we also offer user interfaces and monitoring systems, which are fully integrated into PROCESS-PLC technology.

All about Operating and Monitoring

A great range of alpha-numeric user interfaces are there to serve standardised operator guidance, starting from mere display units with only a few keys, to displays of several lines with a numeric keyboard plus function keys.

If graphics are also to be displayed, the graphic LCD 17 user interfaces present themselves for all relevant possibilities of text and graphic display.

If either display of complex processes and / or data management by PC is required, the following possibilities are provided by the PC-operated VIADUKT monitoring system:

- Graphic process monitoring
- Data transfer to, and from, the controller
- Operating data and error documentation
- Trend graph functions
- Access via JETWay-H to up to 126 controlling devices

F2: Drive Components and their Design

Our service ranges from designing to offering drive series. Let us know the required mechanical data, and we will design the drive for you.

Digital Servo System DIMA with a broad Performance Range

Especially in the area of servo technique, we can offer an integrated solution by a digitised servo system. Its performance ranges from 8 to 100 Ampere. Of course, we will also provide the required drives.

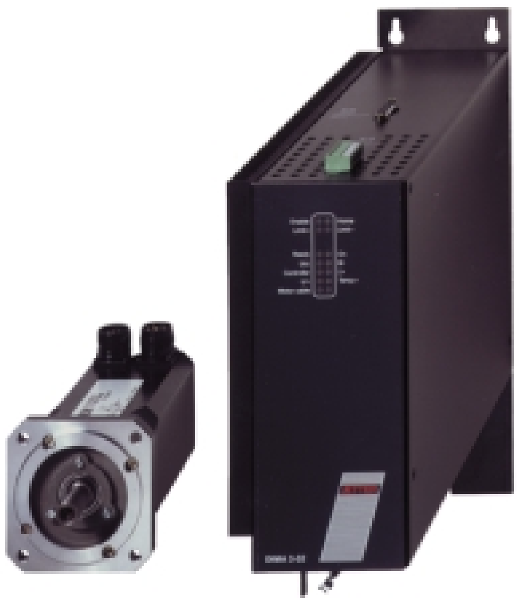
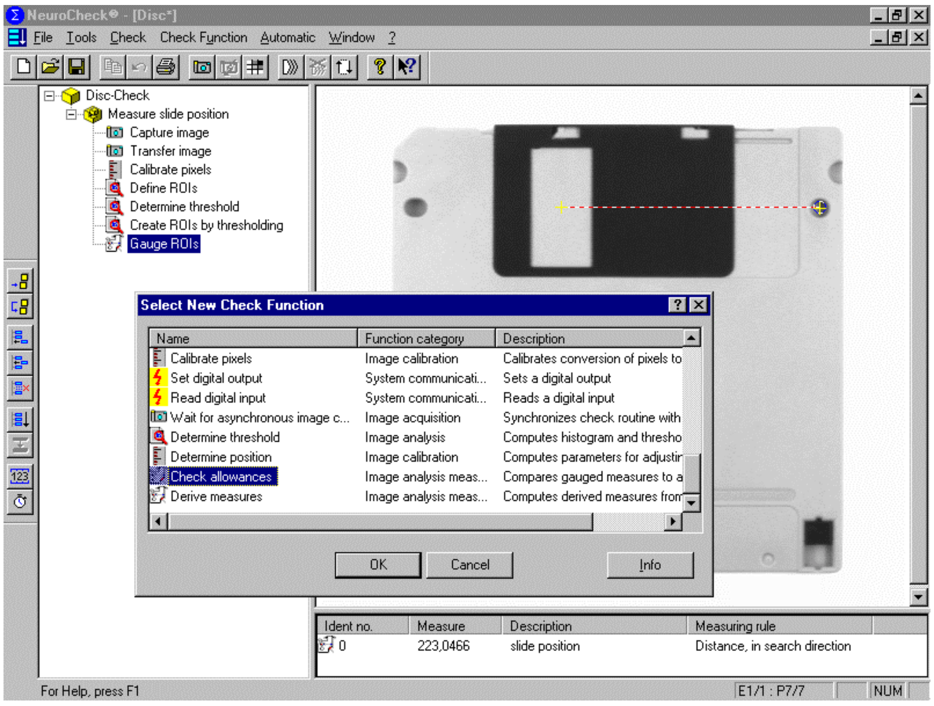


Figure 29: Digital DIMA 32 Motor Control with Servo Motor

F3: Image Processing

In this area, we can also offer you an integrated solution: an image processing system that is open and easy to manage, fully integrated into PROCESS-PLC technology.



Name	Function category	Description
Calibrate pixels	Image calibration	Calibrates conversion of pixels to
Set digital output	System communicati...	Sets a digital output
Read digital input	System communicati...	Reads a digital input
Wait for asynchronous image c...	Image acquisition	Synchronizes check routine with
Determine threshold	Image analysis	Computes histogram and thresho
Determine position	Image calibration	Computes parameters for adjustir
Check allowances	Image analysis meas...	Compares gauged measures to a
Derive measures	Image analysis meas...	Computes derived measures fror

Ident no.	Measure	Description	Measuring rule
0	223,0466	slide position	Distance, in search direction

Figure 30: NEUROCHECK is totally integrated into the PROCESS-PLC and is called-up by SYMPAS like all the other functions.

F4: System Technique

From building control cabinets to programming and setup, we can offer you full service. Yet, it is not our aim to manage a lot of programming projects, but to make available to you the entire know-how about our control systems. If you are looking for a system supplier, though, we would be pleased to offer you this service.

F5: Training

Seminars orientated towards practical experience for beginners and on advanced level

For beginning with PROCESS-PLC programming we offer three-day seminars, which are centred around practical experience. With the help of realistic models, the participants will write and set up exemplary programs building up on each other.

For advanced PROCESS-PLC users we offer workshops, where more complex programs are written.

Please order our seminar program.

Index