DELTA
CPU Module

User's Manual

JETTER GmbH
Gräterstraße 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

JETTER
PROCESS-PLC

Edition 1.0
July 1997

II. DELTA-CPU MODULE AND HOUSING (BASIC UNIT) 2

II. DELTA-CPU Module and Housing (Basic Unit)

1. Safety Instructions

- The PROCESS-PLC DELTA 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 home sensors, protection devices, etc., into the system.

Note:

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

2. Introduction, Configuration

DELTA Maximum Expansion	
Program Space	128 kbyte
User registers 24 Bit	220000 Register nonvolatile RAM
Programming Interface	1
Process Monitoring Interface	1
JETWay-H (Network)	2
JETWay-R (Network)	2
JETWay-CAN	2
Free Programmable Interface RS232/422/485	3
Real Time Clock	1
Digital Inputs 24V	464
Digital Outputs 24V 0,5A	240
Analogue Inputs	184
Analogue Outputs	92
Servo Axis	14
Stepper Motor Axis	21
PID Controller	28

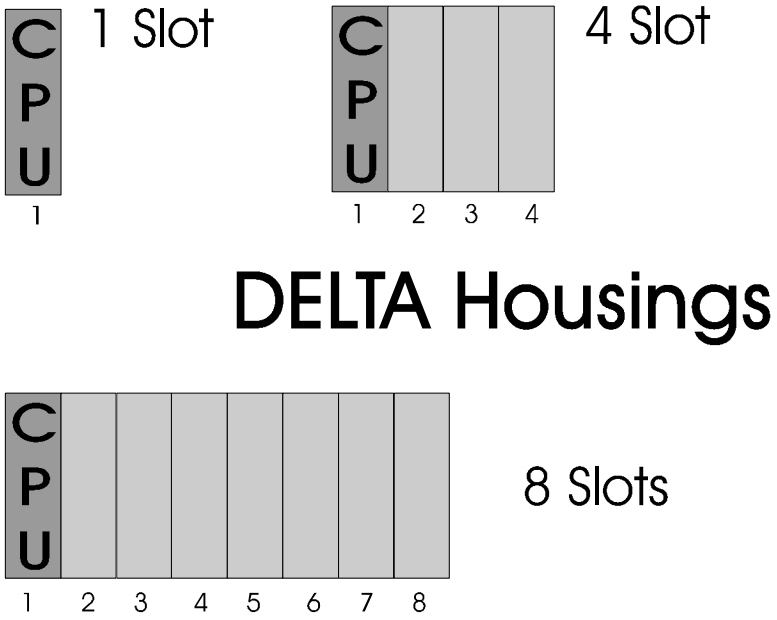
Note:

The maximum expansion relates to the individual function. Two or more functions can not in each case be expanded to the maximum expansion.

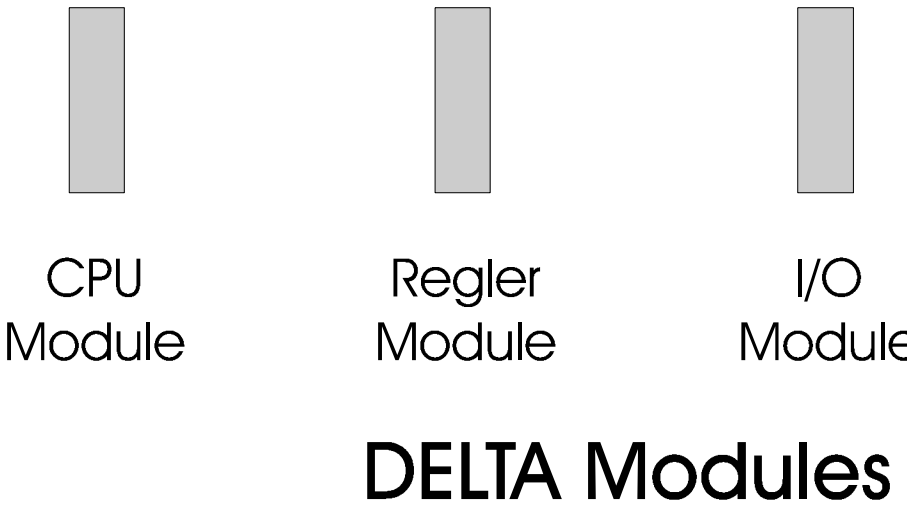
The Housings and Modules

The DELTA base unit cosists of the housing and the CPU module. 3 types of housings are available.

CPU module always in the first slot



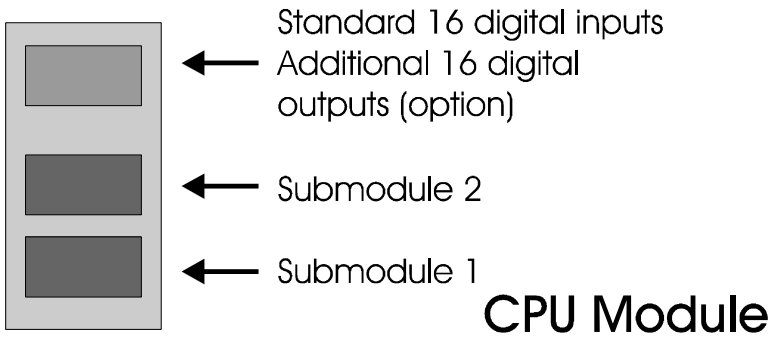
The DELTA modules

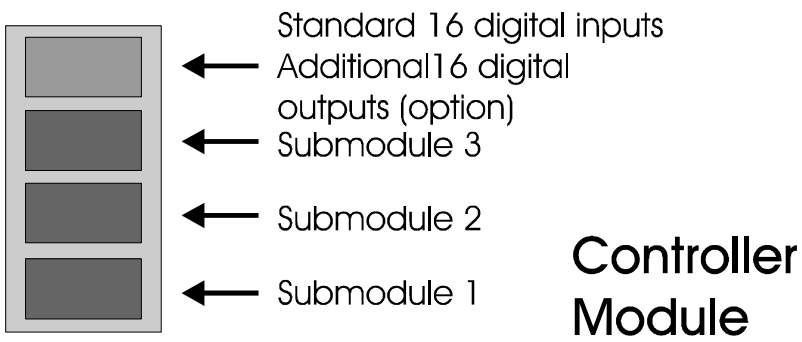


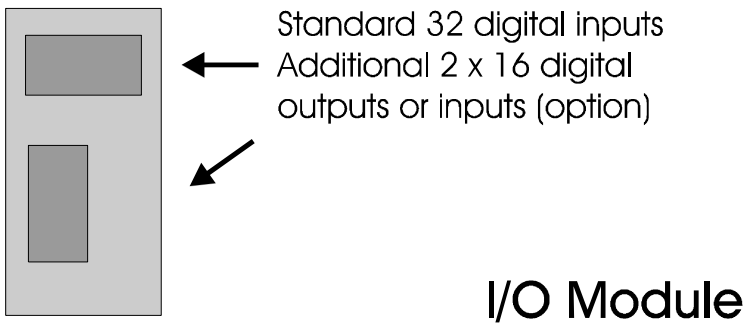
The Modules and Submodules

The CPU module occupies the first slot. Further slots are available for controller, I/O, etc. modules.

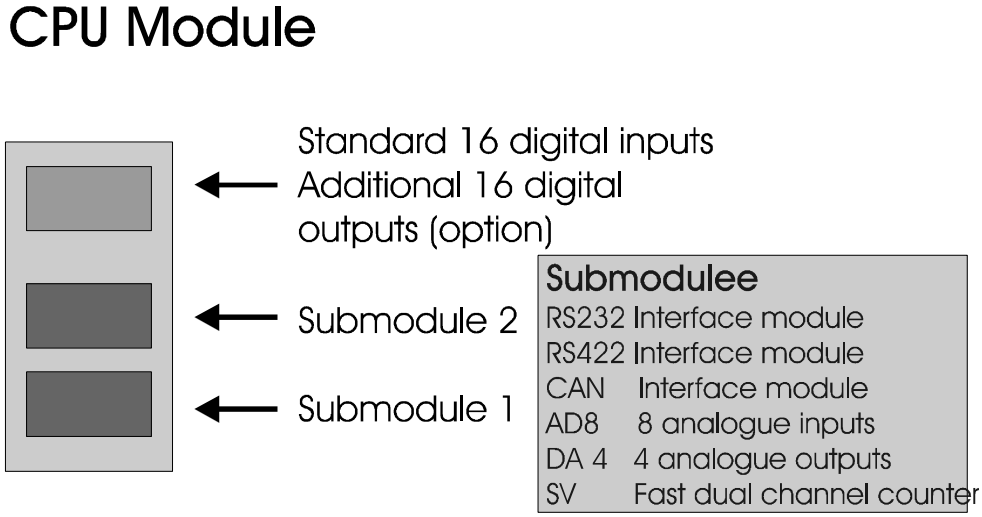
Further submodules are plugged into the DELTA modules



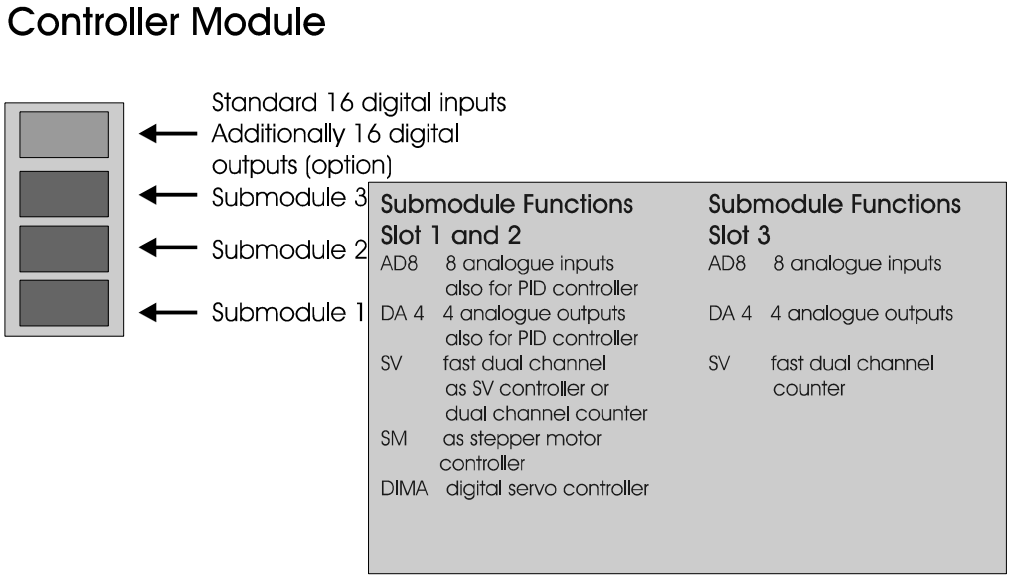




The Submodules of the CPU-Module



The Submodules of the Controller Module



3. General Technical Specifications

Note:

The general technical specifications listed below apply to all PROCESS-PLC 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 IEC 1131-2
ESD	Level ESD-4	according to IEC 1131-2
Housing	Aluminium	

4. Instructions on EMI

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

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.

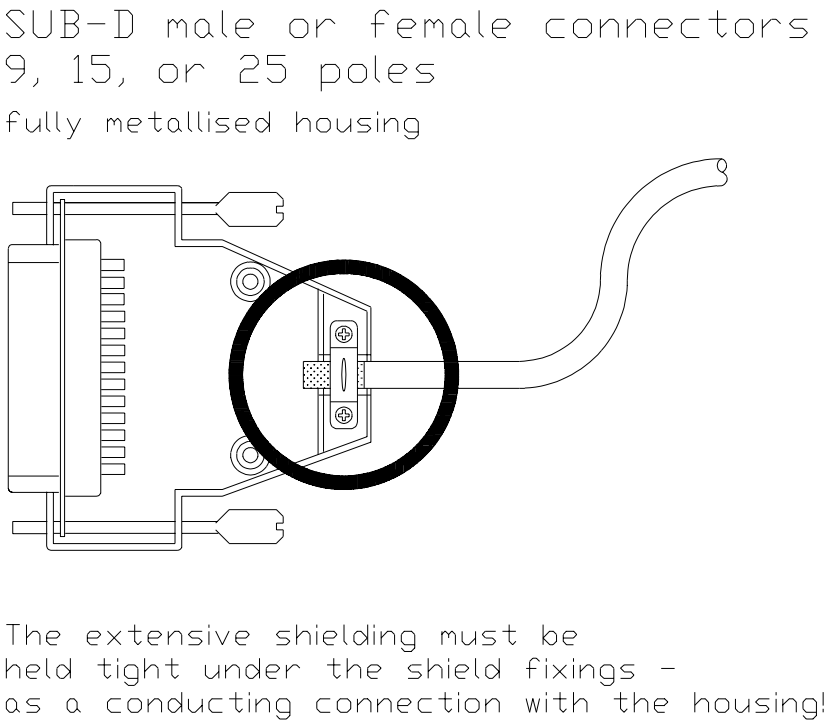


Figure 1: Shielding in Agreement with EMI

5. DELTA Housings (Mechanical Dimensions)

3 types of DELTA housings are available

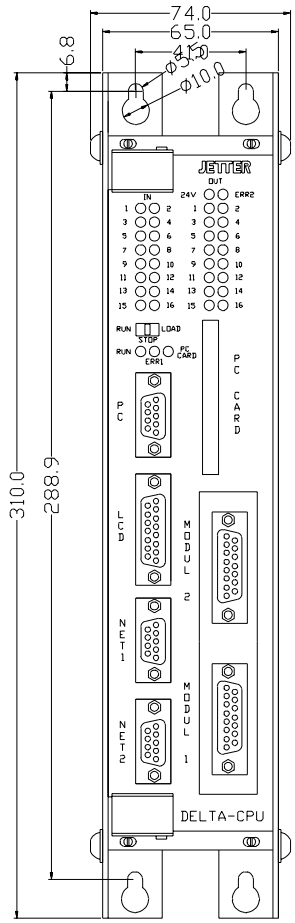
- 1 Slot, CPU Module
- 4 Slot, CPU Module, 3 Expansion Modules
- 8 Slot, CPU-Module, 7 Expansion Modules

Note:

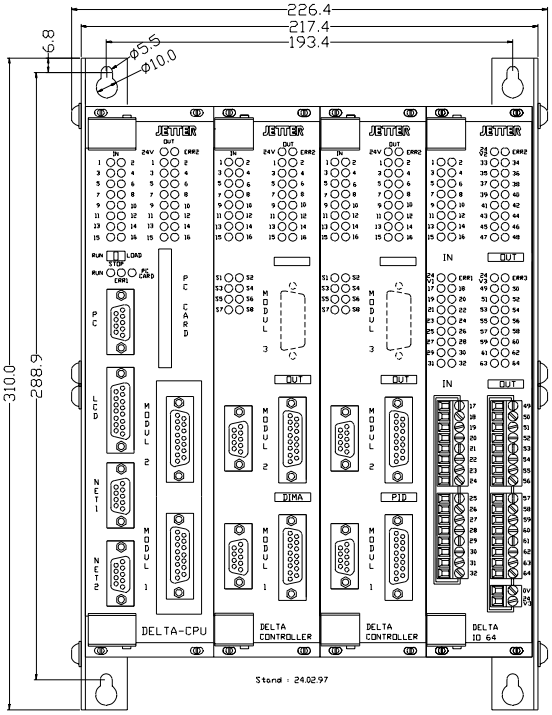
The earthing is placed at the top of the DELTA housing. The earthing has to be connected with the ground.

5.1 Mechanical Dimensions

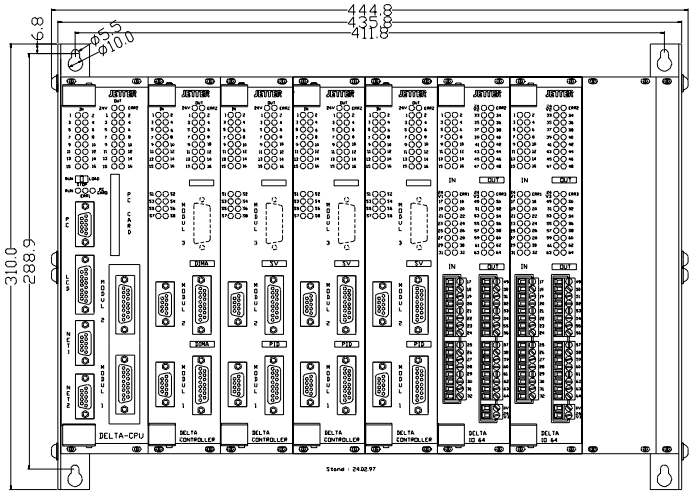
Front view
1 slot
DELTA



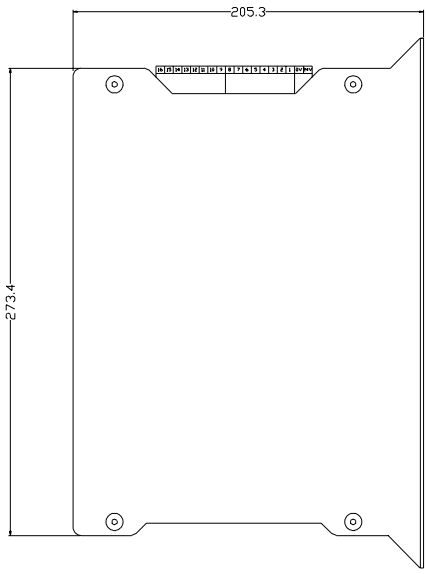
Front view
4 slot
DELTA



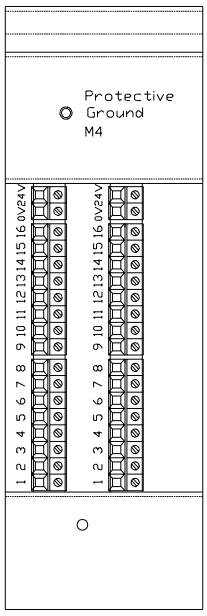
Front view
8 slot
DELTA



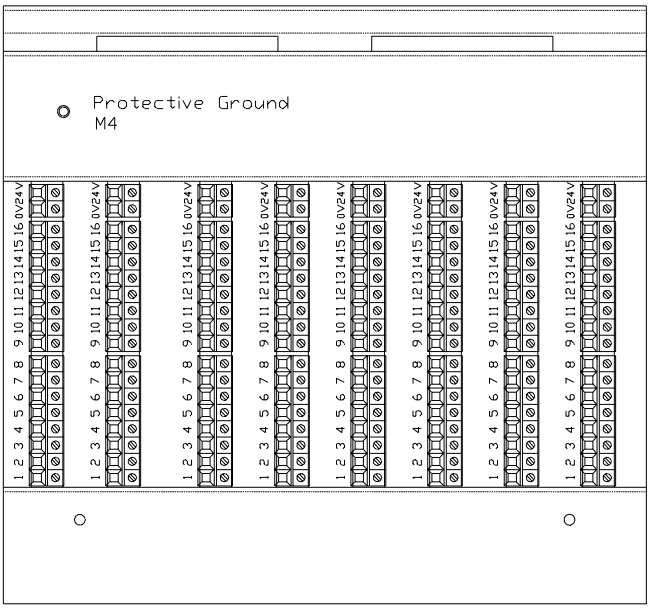
Side view



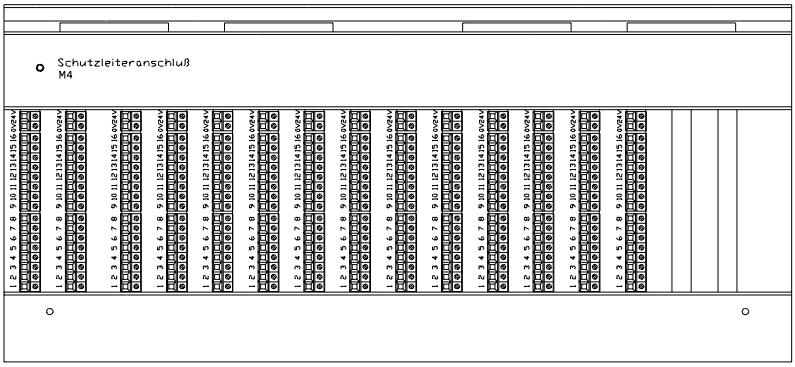
Top view
1 slot
DELTA



Top view
4 slot
DELTA



Top view
8 slot
DELTA



5.2 Slot Numbering

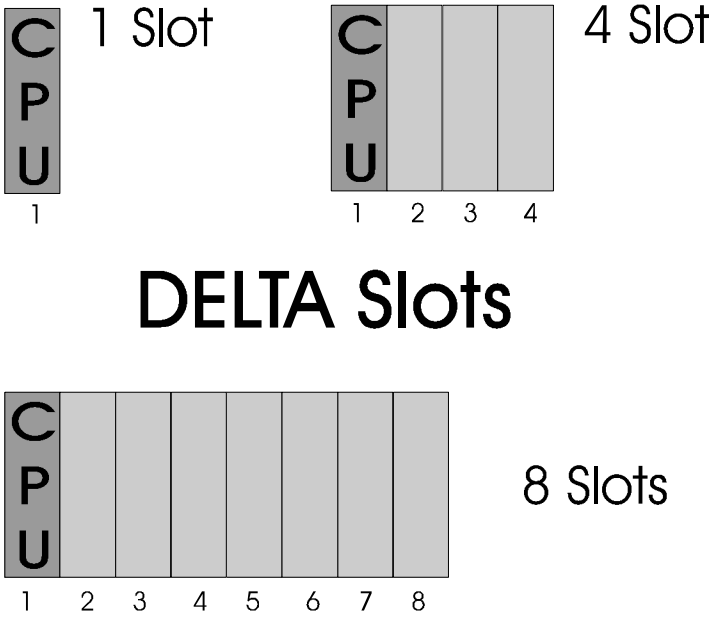
The slots are numbered from the left to the right in increasing order. The number of the left slot is 1 and it is occupied by the CPU module. Expansion modules can be plugged into the slots 2 to 8.

4 DELTA types are available:

DELTA 1

DELTA 4

DELTA 8



6. CPU Module

6.1 Overview and Technical Specifications

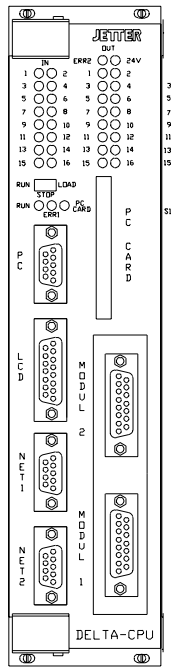


Figure 2: CPU Module

The CPU module contains the microprocessor that executes the user program and controls the functions of the DELTA. The operating system of the DELTA is stored into a Flash-EPROM and can be updated via serial PC interface or PC card.

The CPU module must be plugged into the left slot.

The features of the CPU module:

- CPU, 128kByte program space, 20000 registers
- 1 programming interface
- 1 interface for the user interface
- 1 JETWay interface (network)
- 1 JETWay / free programmable interface
- 2 slots for (for RS232/RS422/CAN interfaces, etc.)
- 16 digital inputs
- 16 digital outputs (optionally)
- Operating system update via programming interface

DELTA CPU Module	
Program Space	128 kByte Flash-EPROM plus RAM (non volatile)
User Registers 24 Bit	20000 non volatile RAM
Data Format	**24 bit integer:** - 8.388.608 ... + 8.388.607 **32 bit floating point:** -10^{15} .. $+10^{15}$
Intermediate Results	32 bit
Number of Flags	2048
Digital Inputs	16 (24 V =)
Digital Outputs	16 pnp 24 V = 0.5 A
Submodule Slot 1	
Submodule Slot 2	
Real Time Clock	1
Free Programmable Serial Interface	RS 232 / RS 485 / RS 422
Programming Interface	RS 232
User Interface and Process Monitoring Interface	RS 232 / RS 422
Fieldbus Interface JETWay 1	RS485
Fieldbus Interface JETWay 2	RS485
Dimensions (H x W x D in mm) 1 Module 4 Module 8 Module	 310.0 x 740 x 205.3 310.0 x 226.4 x 205.3 310.0 x 444.8 x 205.3

Terminals	
Power Supply	plug-in screw terminal
Digital I/O	plug-in screw terminal
Programming Interface	female SUB-D 9-pin
Free Programmable Serial Interface	female SUB-D 9-pin
Fieldbus Interface JETWay	female SUB-D 9-pin
Interface fpr user interfaces and process monitoring	female SUB-D 15-pin

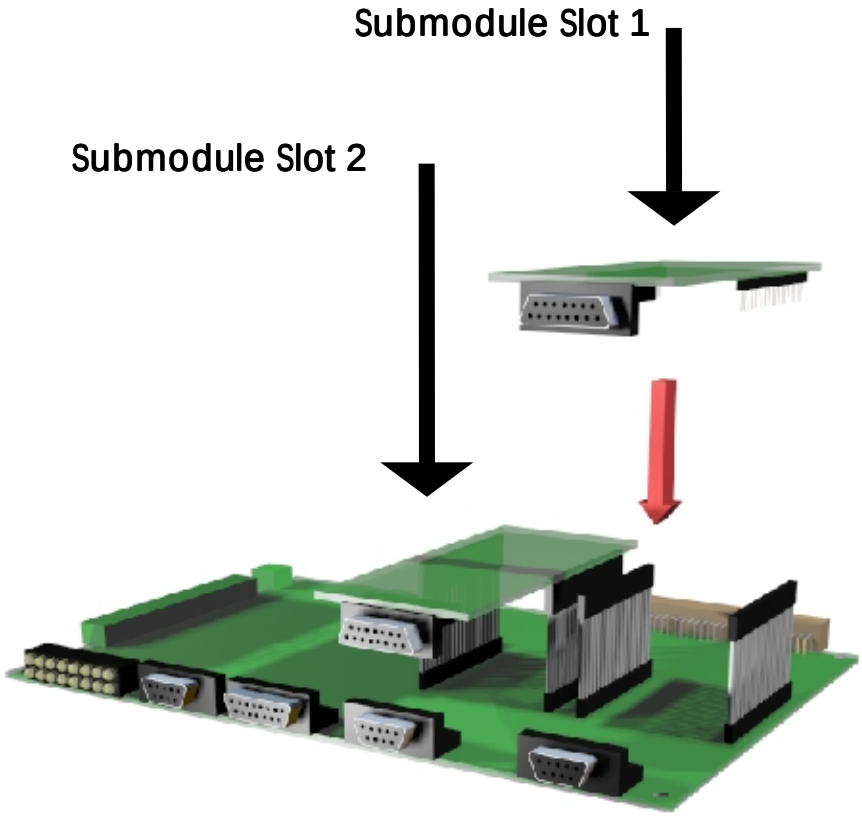
Power Supply	
Requirements Power Supply	20 .. 30 V = ripple < 5% filtered
Power Consumption without digital outputs	ca. 10 Watt (without LCD, option)

Switch, LED´s	
Switch Positions RUN / STOP / LOAD	Switch position STOP: the user program does not start after power on Switch position LOAD: update operating system
LED RUN	Operating system runs
LED ERR1	Operating system error: Error code in Reg 61477

LED IN 1 - 16	24 V at input
LED OUT 1 - 16	output 24 V
LED OUT 24V	external 24V output supply Ok
LED OUT ERR2	Error one or more output driver

6.2 The Submodules of the CPU Module

2 submodules can be plugged into 2 submodule slots of the CPU module.



DELTA CPU Submodules
RS232 Interface Submodule 19,2kBaud
RS422 Interface Submodule 19,2kBaud
CAN Interface Submodule 1MBit/s
AD Module 8 Analogue Inputs -10V .. +10V
DA Module 4 Analogue Outputs -10V .. +10V
SV Module Dual Channel Counter

6.3 Terminals, Specifications

6.3.1 Power Supply

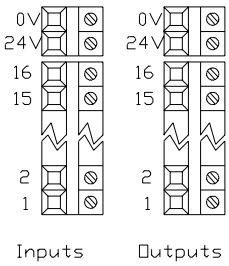


Figure 3: Power supply

The power supply has to meet the following requirements:

Voltage range:	20 VDC.... 30 VDC
Filtered:	ripple 5 %
Power:	ca. 10 W without LCD, Options

Power Supply	
Terminal	**Meaning**
Inputs	
0V	Gnd internal logic
24V	Supply internal logic
Outputs	
0V	Gnd output driver
24V	Supply output driver

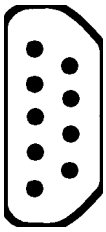
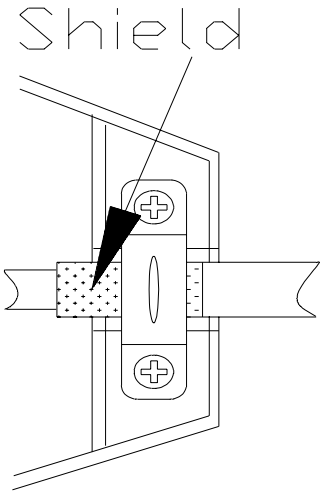
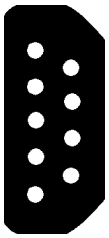
6.3.2 Interfaces

4 connectors for the interfaces are placed on the CPU module.

DELTA CPU Interfaces		
Interface	**Function**	**Specification**
9 pin Sub-D	• Programming • Process Monitoring	RS232 RS232
15 pin Sub-D	• User Interfaces • Process Monitoring • free programmable	RS422 RS232
9 pin Sub-D	• JETWay 1	RS485
9 pin Sub-D	• JETWay 2 • free programmable	RS485 Rs422 RS232

Additional interfaces can be installed by submodules. This submodules can be plugged into the 2 submodule slots of the CPU module.

6.3.2.1 PC Programming Interface (RS232)

Programming Cable (EM-PK)			
PROCESS-PLC			PC
9 pin Sub-D-male **PC** 	RS232 max. Cable length: 15m  **Attach shield at both sides !** **Use metallised housings !**		9 pin Sub-D-female 
Pin	**Signal**		**Pin**
2	TxD	RxD	2
3	RxD	TxD	3
7	Gnd		5
Pin 7 and 8and Pin 1, 4 and 6 have to be bridged on the PC side (COM1, 2)			

The baud rate can be defined in the "Special / Interface" SYMPAS menu.

Baud rate setting relates to program and DA transfer

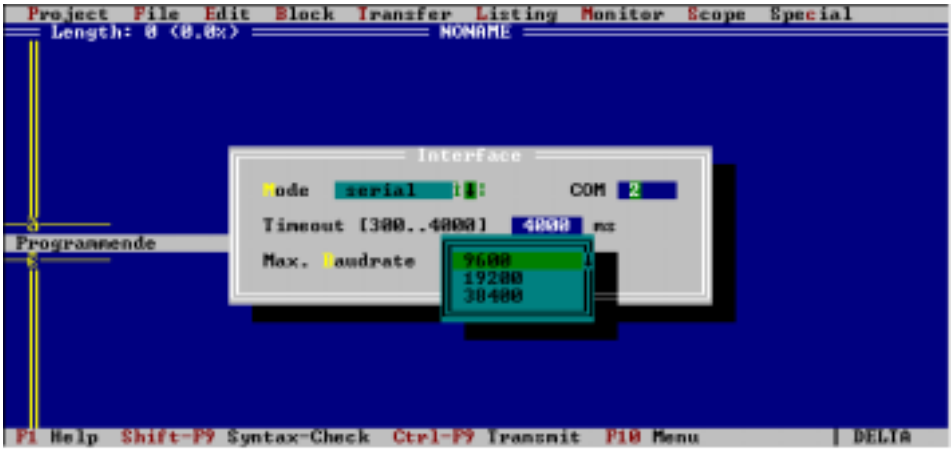


Figure 4: SYMPAS Menu: Special / Interface

Note:

The prefabricated programming 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:	0,25mm²
Connection:	Sub-D, metallised
Shielding:	total, not paired

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

6.3.2.2 PC Programming Interface (JETWay-H)

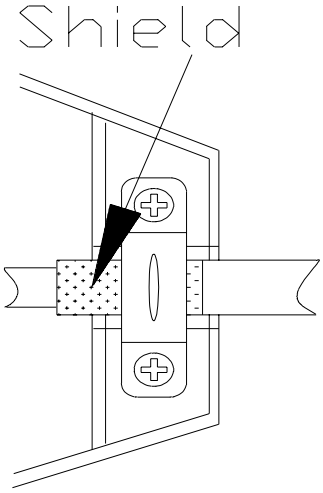
JETWay-H:

126 participants

115 kBaud

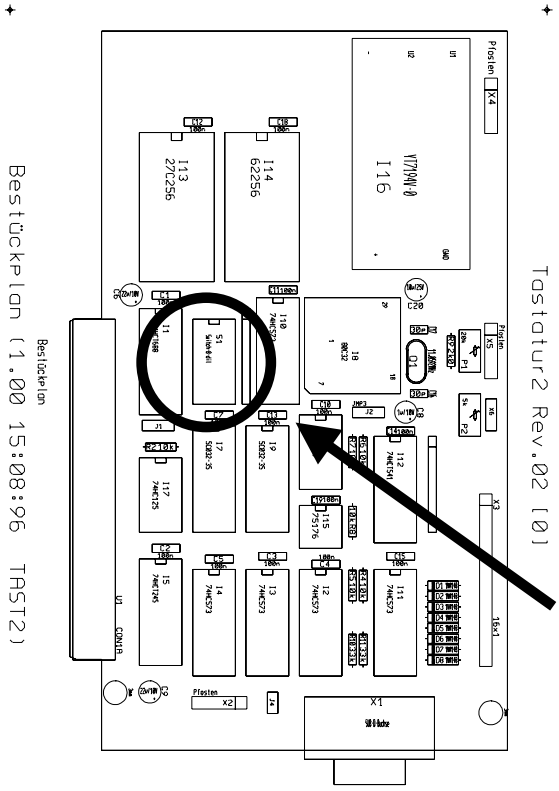
The use of the JETWay-H interface offers following advatages over the RS232 interface:

- Up to 126 PROCESS-PLC can be addressed form one SYMPAS desktop
- Baud rates up to 115kBaud can be realized

JETWay-H Cable		
Connection at DELTA	**Shielding**	**Specification max. Length**
9 pin Sub-D-male **NET1** or **NET2**	 **Attach shield at both sides !** **Use metallised housings !**	RS485 max. cable length: 400m
Pin	**Signal**	**Pin**
7	Gnd	7
8	Data +	8
9	Data -	9

The JETWay-H PC-Card

The connection between SYMPAS and up to 126 PROCESS-PLC can be realized with the plug-in PC-card shown below.



Here are the DIL switches for port address setting. Default is 340h.

Figure 5: JETWay-H plug-in card for PC

AUTOEXEC.BAT

Insert following line into your AUTOEXEC.BAT (given you use the default setting):

SET JETWAY_PORT=340h

DIL Switch

Select various port addresses with help of the DIL switches of the JETWay-H PC-card.

DIL Switches of the JETWay-H PC-Card						
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						

Change the line in the AUTOEXEC.BAT corresponding to the DIL switch settings:

SET JETWAY_PORT=x

Select between the programming interface via RS232 or JETWay-H with the "Special / Interface" SYMPAS menu.

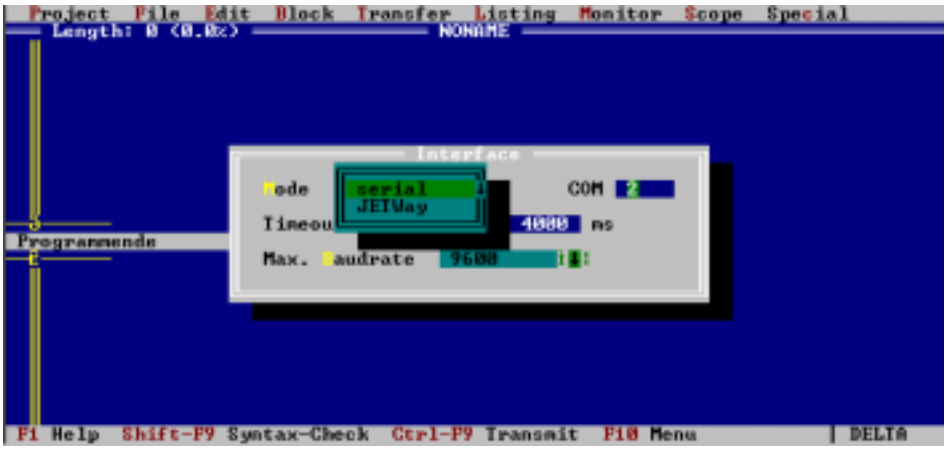


Figure 6: SYMPAS Menu: Special / Interface

Note:

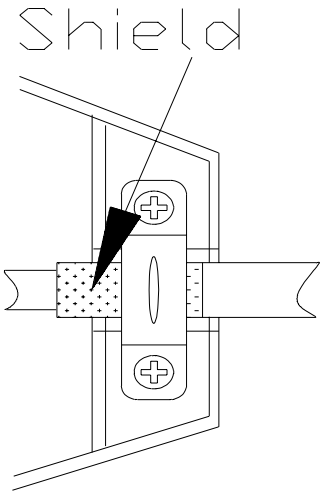
If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	3
Diameter:	0,25mm²
Connection:	Sub-D, metallised
Shielding:	total, not paired

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

6.3.2.3 Network Interface (JETWay-R)

JETWay-R establish connection between the PROCESS-PLC and/or PROCESS-PLC and remote I/O, valve blocks, etc.

JETWay-R Cable		
Connection on DELTA side	**Shielding**	**Specification max. Length**
9 pin Sub-D-male **NET1** or **NET2**	 **Attach shield at both sides !** **Use metallised housings !**	RS485 max. cable length: 400m
Pin	**Signal**	**Remark**
7	Gnd	
8	Data +	
9	Data -	

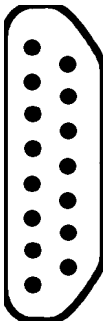
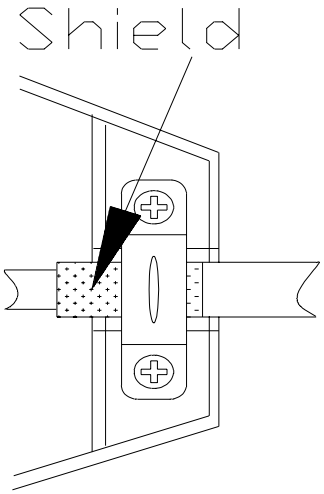
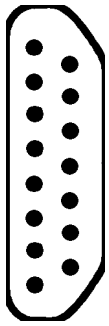
Note:

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

Number of wires:	3
Diameter:	0,25mm²
Connection:	Sub-D, metallised
Shielding:	total, not paired

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

6.3.2.4 User Interface Connections

User Interface Cable (DK-422)			
PROCESS-PLC			User Interface
15 pin Sub-D-male **LCD** 	RS422 max. cable length: 400m  **Attach shield at both sides ! Use metallised housings !**		15 pin Sub-D-male 
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 programming cable DK-422 can be obtained from JETTER.
If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	6
Diameter:	0,25mm²
Connection:	Sub-D, metallised
Shielding:	total, not paired

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

6.3.2.5 Process Monitoring Interface (RS232)

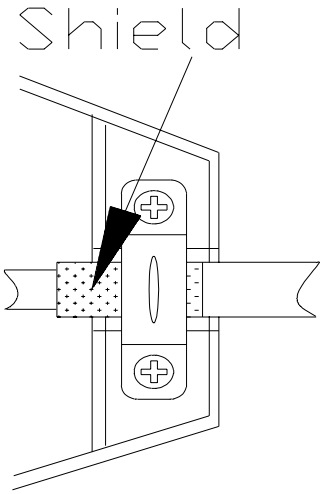
Possible interfaces:

The process monitoring system JETVisu can be connected via 2 different cables with the PROCESS-PLC.

RS232

or

JETWay-H

JETLink Cable RS232			
Connection			**JETVisu**
9 pin Sub-D-male **PC** or 15 pin Sub-D-male **LCD**	 **Attach shield at both sides !** **Use metallised housings !**		RS232 max. cable length: 15m
Pin	**Signal**		**Pin**
2	TxD	RxD	2
3	RxD	TxD	3
7	Gnd		7
Pin 7 and 8and Pin 1, 4 and 6 have to be bridged on the PC side (COM1, 2)			

6.3.2.6 Process Monitoring Interface (JETWay-H)

JETWay-H:

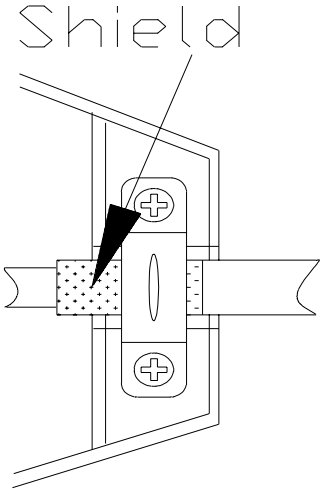
The use of the JETWay-H interface offers following advantages over the RS232 interface:

126 participants

- Up to 126 PROCESS-PLC can be addressed from one JETVisu process monitoring system

115 kBaud

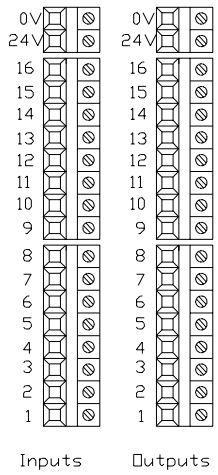
- Baud rates up to 115kBaud can be realized

JETWay-H Cable		
Connection on DELTA side	**Shielding**	**Specification max. Length**
9 pin Sub-D-male **NET1, NET2**	 **Attach shield at both sides !** **Use metallised housings !**	RS485 max. cable length: 400m
Pin	**Signal**	**Remark**
7	Gnd	
8	Data +	
9	Data -	

JETWay-H PC-card necessary (plugged into VIADUKT)

The JETWay-H PC-card has to be plugged into the VIADUKT or PC that executes JETVisu. All PROCESS-PLC which are monitored are connected with this PC-card. See Chapter *6.3.2.2 PC Programming Interface (JETWay-H)*

6.3.4 Digital Inputs



16 terminals for digital inputs (24VDC) have been provided on the upper side of the controller housing. The 0V signal is connected to the control cabinet ground (Gnd).

Technical Specifications Digital Inputs	
Number of Inputs	16
Rated Input Voltage	24 VDC
Voltage Range	15 .. 27 V
Input Current	approx. 8 mA
Input Resistance	3,0 kΩ
Input Delay	ca. 3ms
Signal Voltage ON	min. 15 V
Signal Voltage OFF	max. 10 V
Potential Isolation	no

Input Numbering on the CPU Module	
Input	**Number**
Input 1	101
Input 2	102
...	...
Input 16	116

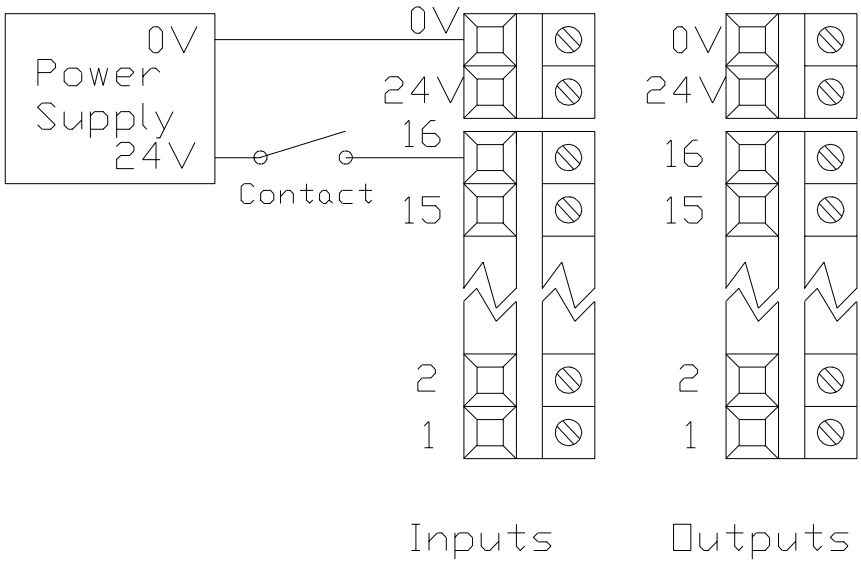


Figure 7: Digital input wiring

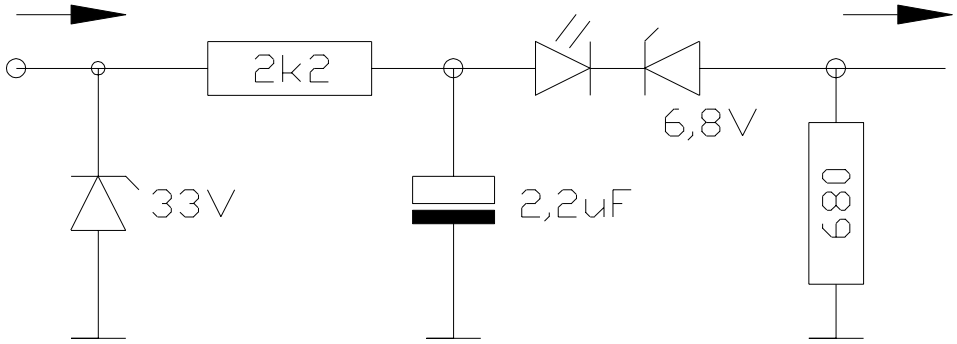
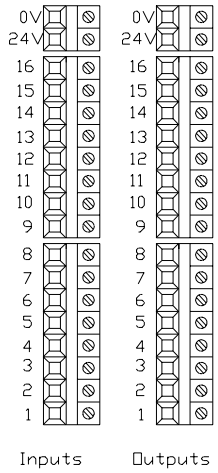


Figure 8: Internal circuit of the digital inputs

6.3.5 Digital Outputs



16 terminals for digital outputs (24VDC) have been provided on the upper side of the controller housing. The 0V signal is connected to the control cabinet ground (Gnd).

Technical Specifications digital Outputs	
Number of Outputs	16
Output Type	Transistor, pnp
Rated Voltage	24 VDC
Voltage Range	20 .. 30 V
Load Current	max. 0,5 A / output
Potential isolation	no
Protection Circiut	Overload, overvoltage, overtemperature
Protection Inductive Loads	yes
Signal Voltage ON	typ. V_{Supply} - 0,5 V

Output Numbering on the CPU Module	
Output	**Number**
Output 1	101
Output 2	102
...	...
Output 16	116

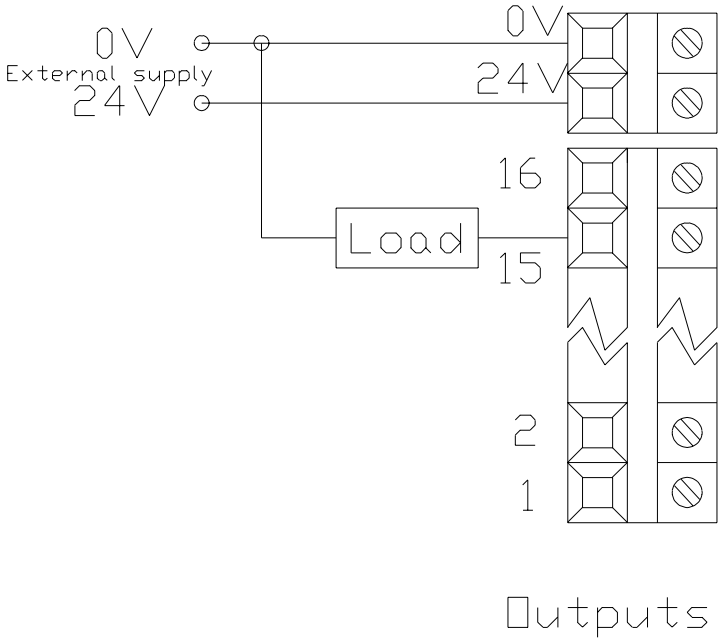


Figure 9: Digital output wiring

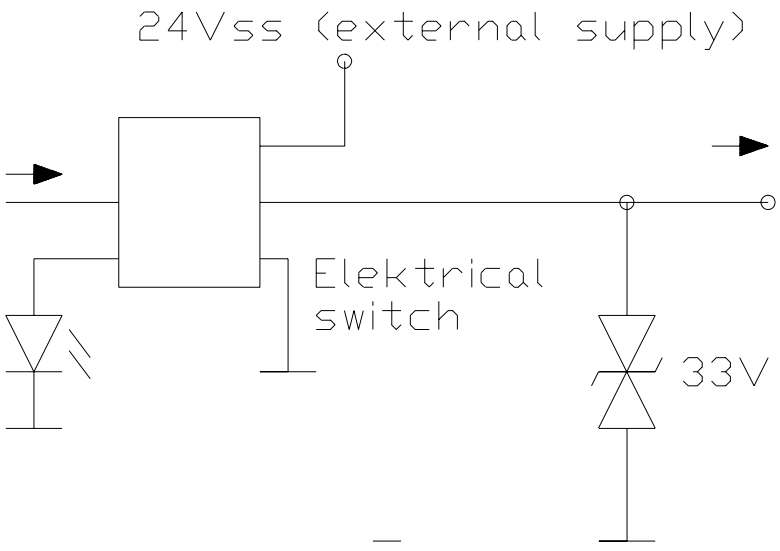
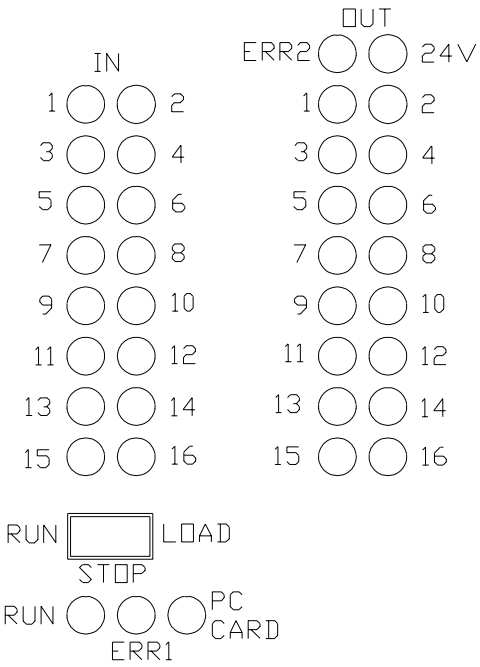


Figure 10: Internal circuit of the digital outputs

6.4 The LED



The LED indicate the states of the digital inputs and outputs as well as the operating system.

Figure 11: LED of the CPU modules

LED of the Digital Inputs on the CPU Module	
LED	**Meaning**
IN 1	lights at 24V
...	
IN 16	-"-

LED of the Digital Outputs on the CPU Module	
LED	**Meaning**
OUT 1	Lights at 24V
...	
OUT 16	-"-
24 V	External power supply of the outputs OK
ERR2	Error of 1 or more output drivers

LED of the Operating System on the CPU Module	
LED	**Bedeutung**
RUN	Operating system runs
ERR1	Operating system error Error code in register 61477

6.5 The Switch

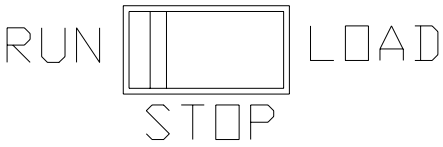


Figure 12: Switch of the CPU module

Switch on the CPU Module	
Position	**Meaning**
RUN	User program **runs** after power-on of the controller
STOP	User program is **stopped** after power-on of the controller
LOAD	User program stopped. **Operating system update** possible

Note:

The switch is evaluated after power-on of the controller. Switching during controller operation is not recognized.

6.6 Addressing / Description Inputs, Outputs, Flags, Registers (Programming)

Programming of register, flags, inputs, outputs

This chapter describes the access to the DELTA registers, inputs, outputs and flags.

The numbering of the registers, secial registers, flags, special flags, inputs and outputs is listed in tables.

Following instructions are used to access registers, flags, inputs and outputs.

REGISTER_LOAD, REG	access to registers
FLAG	access to flags
INPUT	access to inputs
OUTPUT	access to outputs

Example:

```
REGISTER_LOAD [100 with 1234] ;loads register 100
                              ;with 1234
REG 100 = REG 300 + REG 200   ;adds the contents
                              ;of register 300
                              ;register 200 and
                              ;stores the result
                              ;in register 100
```

```
WHEN                          ;When flag 10
    FLAG 10                   ;active continue
  THEN                        ;task execution
  ...


WHEN                          ;When input 101
    IN 101                    ;active then
  THEN                        ;set output 105
    OUT 105
```

6.6.1 Addressing of the Digital Inputs and Outputs of a CPU Module

Input Numbering on the CPU Module	
Input	**Number**
Inputs 1	101
Inputs 2	102
...	...
Inputs 16	116

Output Numbering on the CPU Module	
Output	**Number**
Output 1	101
Output 2	102
...	...
Output 16	116

6.6.2 Access of Flags

6.6.2.1 User Flags

Flag 1 to 2047 are available user flags

Flag 1 to 2047 are available user flags. Some of this flags are overlayed with registers 0 to 74. Thus flag ranges can be accessed via registers. Bit operations can be realized with the W-AND, W-OR and W-XOR instructions.

Overlay of Registers and Flags

Register	Flag	Remark
0 to 74	1 to 255	free
	256 to 2047	**Overlay**
	2048 to 2303	Special flags

Example:

Overlay User Flags - Registers Exemplary for Register 0									
Bitno	0	1	2	3	4	...	21	22	23
Flag	256	257	258	259	260	...	277	278	279

Programming with Flags

Example 1:

The execution of a process is to start, if the start button is pressed and the automatic mode is released by by setting of the corresponding flag (for example in an other task).

```
WHEN
    IN iStartKey
    FLAG fAutomatic
  THEN
    ...
```

Example 2:

A flag in the main task starts execution of a second task, the automatic task.

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

6.6.2.2 Special Flags

The PROCESS-PLC operating system provides various special flags that allow control or modification of functions. The following table gives an overview of the special flags, structured by functions and with the cross-reference to chapters that describe the functions more comprehensively.

Note:

Basically a set flag means the activation of the function. Exceptions will be specially referred to.

Special Flags User Interface Keys / LED

Control of the User Interface LED

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 the User Unterface 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

Special Function Keys			
<-	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
Numerical 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

Special Flags

Special Flag	Function	Cross-Reference
(Error) Messages via Special Flags		
2104	timeout during last slave access	

2105	at minimum 1 timeout since reset during slave register access	
2110	timeout during last network access (JETWay)	
2111	at minimum 1 timeout since reset during network access	
2136	invalid label for GOTO or CALL indirect	
2137	current instruction would cause stack overflow; task was breaked	
2138	current instruction would cause stack underflow; task was breaked	
2139	no user program or CRC error	
2140	OP code error	
2144	output error message	
2145	real time clock identified	
2146	battery real time clock OK	
2147	battery register RAM almost empty	
Task Control		
2056	taskswitch after (reg 61804) ms	
2057	taskswitch if GOTO instruction	
2058	taskswitch if not fullfilled IF instruction	
2112	PC and user interface after each task	
Network Control with Special Flags		
2152	multimaster mode network 1	
2153	multimaster mode network 2	

Arithmetic		
2048	use real instead of integer for division	0 = on
2049	real mode	
User Interface Control		
2051	USER_INPUT instruction is active	
2053	USER_INPUT breaked by timeout	
User Interface Restrictions		
2052	user interface input locked	
2054	(ENTER) switches not to monitor mode	
2096	no register display	0 = locked
2097	no flag display	0 = locked
2098	no output display	0 = locked
2099	no input display	0 = locked
2100	no register input	0 = locked
2101	no flag input	0 = locked
2102	no output input	0 = locked
2103	no continuous input display	

6.6.3. Register Description (The DELTA Data)

6.6.3.1 User Registers

User registers: 0 to 20479

20480 user register are available in the register range of 0 to 20479. They serve as working registers, buffers for various variables, parameters, etc.

The registers have a width of 24 bit and a value range of -8,388,608 to +8,388,607.

For example registers can be loaded with the REGISTER_LOAD instruction.

Note:

The content of the 20480 user registers remains after power-off of the controller.

Figure 14:REGISTER_LOAD with numerical parameters

Figure 13: REGISTER_LOAD with symbolic parameters

Programming with Registers

The instruction

REGISTER_LOAD [x with a]

loads values from one register into another.

Call-up in SYMPAS with (L) (R)

Description:

x defines the number of the register into which the number **a** is to be written.

Indirect and Double Indirect Adressing

Indirect level with (SPACE) key 1 time, 2 times

The **x** and the **a** of the example above need not to be a number. They can also be replaced by a register (indirect addressing). The indirect level can be obtained by pressing the (SPACE) key. In this case **R** is set before the register number.
The value **a** is written into the register with the number **y** if **R(y)** is used instead of **x**.
If **R(b)** is used instead of **a** then not the value itself but the content of the register with the number **b** is written into the register which number is specified by **R(y).**

If you now enter **RR** (2 times space key) and then the number **(b)** instead of **a**

REGISTER_LOAD [x with RR(b)]

the following happens:

- first the content of the register with the number **b** is read
- this value is now used as **register number**; from the register with this number the content is read and stored into register **x**

Indirect and double indirect addressing of registers

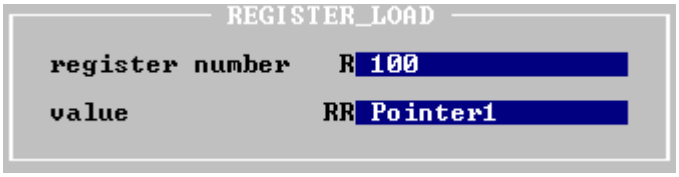


Figure 15: The indirect levels R and RR can be entered with (SPACE) or (CTRL) (R)

Example:

1) Load a number into a register

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

The value 1280 is stored into register `rNewPosition`.

2) Copy one register into another

```
REGISTER_LOAD [ rVoltage with R(rVoltage1)]
```

The value of register `rVoltage1` is copied into register `rVoltage`.

3a) Loading by using double indirect addressing

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

The value which is in the register with the number that is content of register `rU_Pointer` is stored into register `rVoltage`

3b) Example of double indirect addressing:

Registers	Value
REG 64	111
REG 111	70035
REG 150	11
REG 11	any

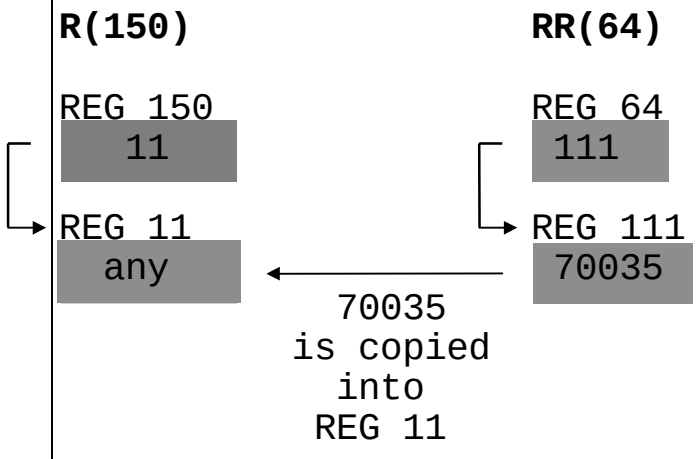
following instruction is executed with this register assignment:

LADE_REGISTER [R(150) mit RR(64)]

Following register values result:

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

Diagram:



Calculating with Registers

Instructions for calculating:

REG <RegNo>

REGZERO <RegNo>

REGDEC <RegNo>

REGINC <RegNo>

This 4 instructions allow indirect addressing. For instance RegNo can be R100. This means the instruction effects the content of the register that is specified by the number in REG 100.

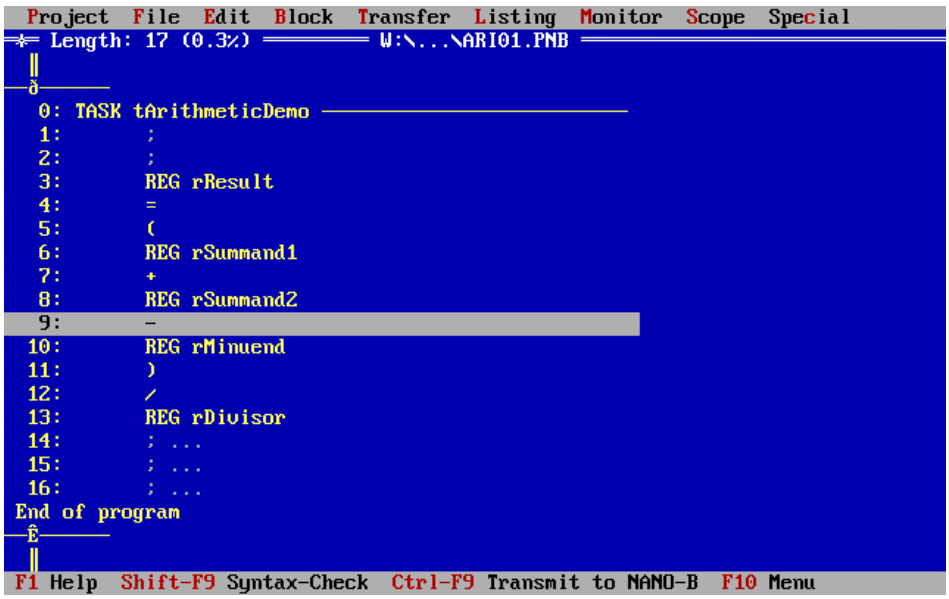


Figure 37: Simple Example on Register Arithmetic

REG

Call-up in SYMPAS with (R) (E)

This instruction directly accesses the content of a register and is equivalent to a variable. In an output instruction a value is assigned to the register that stands at the left side of the equal sign. In an input condition the register content is read. Accesses to registers at the right side of the equal sign always read the register contents.

Example:

Example for an output instruction

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

This example shows an **assignment (output instruction** introduced by THEN). Thereby register 105 is read and its content is multiplied with 25. The result of this operation is stored into register 1. The content of register 105 remains unchanged.

Example for an input condition

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

In this case the expression `REG 1 = REG 105 * 25` is not placed in an output instruction but in an **input condition**. Therefore the value of register 1 is not modified but compared with the product of `REG 105 * 25`.

The REGZERO condition sets a register to 0 or queries if a register is 0:

REGZERO <RegNo>

Call-up in SYMPAS with (R) (N)

This instruction has following meaning used in an input condition (after IF or WHEN).

Example:

```
IF
    REGZERO 49
  THEN
```

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

Both program parts have the same function. On the right the comparison is executed as common arithmetic comparison; on the left the REGZERO instruction is used (advantages: faster execution, less code).

The instructions

REGDEC REGINC

Call-up in SYMPAS with (R) (D) resp. (R) (I)

This registers are used to increment or decrement a register by 1. These instructions are often used in loops, for increment or decrement of counters and pointers.

Example:

1a)

```
THEN
  REGDEC 100
```

1b)

```
THEN
  REG 100
  =
  REG 100
  -
  1
```

Both program parts have the same function. Both decrement the content of register 100 by 1.

Both examples have the same function

2a)

```
THEN
  REGINC 88
```

2b)

```
THEN
  REG 88
  =
  REG 88
  +
  1
```

Also here both program parts have the same effect. Register 88 is incremented by 1.

Example: loop

3a)

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

A loop can be realized this way that executes a certain number of loops. The counter register is decremeted by 1 each loop and compared with 0. If it is 0 nothing is done at the first THEN and the program flow continues to the second THEN. If it is not yet 0 the ELSE instruction branches to the loop top.

3b)

```
        REGISTER_LOAD [ 1 with 10]
LABEL   55
        ...
        REGDEC 1
   IF
        REG 1                ;≠0 -> true
     THEN                    ;=0 -> false
       GOTO 55
     THEN
```

6.6.3.2 Special registers

Special Registers Overview		
Register no	**Function**	**1) Value range 2) Reset value 3) Cross-reference**
Operating System- (Error-) Messages		
61473	Operating system error (Special flag 2104 .. 2111) Bit0 = 0 no error Bit0 = 1 timeout slave register-access (2104) Bit1 = 0 no error Bit1 = 1 timeout slave register access since reset (2105) Bit3 = 0 no error Bit3 = 1 SPECIALFUNCTION1: more than 99 parameters (2107) Bit6 = 0 no error Bit6 = 1 timeout last network access (2110) Bit7 = 0 no error Bit7 = 1 network-timeout since reset (2111)	1) 0 .. 255 2) 0
61477	Operating system error (Special flag 2136 to 2143) if 61477‡0 -> ERR-LED on Bit0 = 1 illegal destination for GOTO or SUBROUTINE task was breaked (2136) Bit1 = 1 current instruction would cause stack overflow, task was breaked (2137) Bit2 = 1 current instruction would cause stack underflow, task was breaked (2138) Bit3 = 1 no user program or CRC error (2139) Bit4 = 1 OP code-error (2140)	1) 0 .. 255 2) 0 3)
61478	Error, messages operating	1) 0 .. 255

	system Bit0 = 1 Error digital output (2144) Bit1 = 1 real time clock identified (2145) Bit2 = 1 battery real time clock Ok (2146) Bit3 = 1 battery register RAM soon empty (2147)	2) 0
61530	Task number of OPC error	1) 0 .. 255 2) not defined
61952	User program runtime in seconds	1) -8388608 .. 8388607 2) 0 3)
61953	Controller run time in seconds	1) -8388608 .. 8388607 2) 0 3)
61954	Controller runtime in time base units	1) -8388608 .. 8388607 2) 0
61956	Controller runtime since reset in ms	1) -8388608 .. 8388607 2) 0 3)
62977	Operating system version * 100	1) 0 .. 65535 2) 0 3)
Task Control		
61449	Priority task	1) 0 .. 255 2) 255 = no priority task 3)
61467	Taskswitch conditions (Special flags 2056 to 2063) Taskswitch always if o DELAY o USER_INPUT o WHEN (not fullfilled) and also if o Flag 2056 AND taskswitch timeout (61804) o Flag 2057 AND GOTO o Flag 2058 AND IF (not fullfilled) Bit0 = flag 2056 Bit0 = 0 no taskswitch if timeout Bit0 = 1 taskswitch if timeout Bit1 = flag 2057 Bit1 = 0 no taskswitch if GOTO Bit1 = 1 taskswitch if GOTO Bit2 = flag 2058 Bit2 = 0 no taskswitch if IF (not fullfilled) Bit2 = 1 taskswitch if IF (not fullfilled)	1) 0 .. 255 2) 3 3)

61474	Multitasking control (Special flag 2112 .. 2119) Bit0=0 SYMPAS, LCD before TASK0 (default) Bit0=1 SYMPAS, LCD before each Task (2112) Bit5=0 N-SEND-REGISTER, N-GET-REGISTER are interrupted Bit5=1 N-SEND-REGISTER, N-GET-REGISTER are not interrupted (2117) (default)	1) 0 .. 255 2)
61531 .. 61562	Task status: 255 = Task runs 254 = DELAY 253 = USER_INPUT 250 = WHEN_MAX 1 = TASKBREAK 0 = breaked	1) 0 .. 255 2) Status 3) SYMPAS: index window
61610	Highest task number	1) 0 .. 255 2) 0 3)
61773	Minimum cycle time (ms)	1) 0 .. 65535 2) 0
61774	Maximum cycle time (ms)	1) 0 .. 65535 2) 0
61777	Cycle time of all task (ms)	1) 0 .. 65535 2) 0
61804	Task timeout time (ms)	1) 0 .. 255 2) 0
User Interface Control (LCD-Display)		
61448	Display language, 0 = german, 1 = english	1) 0 .. 255 2) Depends on the user program 3)
61451	Field width for floating point register display	1) 0 .. 255 2) 8 3)
61452	Number of decimal places	1) 0 .. 255 2) 4 3)
61453	Field width for integer display	1) 0 .. 255 2) 1 3)
61454	Flush left number display Flag 2055=0 leading blanks Flag 2055=1 leading zeros	1) 0 .. 255 2) 0 3)
61455	Field width USER_INPUT	1) 0 .. 255 2) 8 3)
61461	'Delete line' character	1) 0 .. 255

		2) 36 3)
61462	'Clear display' character	1) 0 .. 255 2) 95 3)
61472	Restrictions for monitor functions 0 = disable, 1 = enable (Flag 2096 .. 2103) Bit0=0 R, I/O key without register number input Bit0=1 R, I/O key with register number input Bit1=0 R, I/O key without flag number input Bit1=1 R, I/O key with flag number input Bit2=0 R, I/O key without output number input Bit2=1 R, I/O key with output number input Bit3=0 R, I/O key without input number input Bit3=1 R, I/O key with output number input Bit4=0 = key changes no register contents Bit4=1 = key changes register contents Bit5=0 = key changes no flags Bit5=1 = key changes flags Bit6=0 = key changes no outputs Bit6=1 = key changes outputs Bit7=0 = key does not access to inputs Bit7=1 = key does access to inputs	1) 0 .. 255 2) 0 3)
61480 .. 61487	User interface keys (Special flags 2169 .. 2223)	1) 0 .. 255 2) Key status 3) See special registers
61488 .. 61489	LED user interfaces (Special flags)	1) 0 .. 255 2) 0 3) See special registers
61648	Indirect cursor position cp=0 for DISPLAY_TEXT und DISPLAY_REG	1) 0 .. 255 2) 0 3)
61649	Indirect cursor position cp=0 for USER_INPUT	1) 0 .. 255 2) 0 3)
61653	Display time for monitor functions (s)	1) 0 .. 65535 2) 3 3)
61683	Text selection for DISPLAY_TEXT_2	1) 0 .. 255 2) 0

	0 = text 1, 1 = text 2	3)
61967	First register changeable by user interface - range 1	1) -8388608 .. +8388607 2) 0 3)
61968	Last register changeable by user interface - range 1	1) -8388608 .. +8388607 2) +8388607 3)
61969	First register changeable by user interface - range 2	1) -8388608 .. +8388607 2) 0 3)
61970	Last register changeable by user interface - range 2	1) -8388608 .. +8388607 2) +8388607 3)
61971	First register changeable by user interface - range 3	1) -8388608 .. +8388607 2) 0 3)
61972	Last register changeable by user interface - range 3	1) -8388608 .. +8388607 2) +8388607 3)
61973	First flag changeable by the user interface	1) 0 .. 65535 2) 0 3)
61974	Last flag changeable by the user interface	1) 0 .. 65535 2) 65535 3)
61818	Max. time for USER_INPUT default = 0 (disabled)	1) 0 .. 65535 2) 0 3)
61825	Number character per line	1) 0 .. 255 2) 24 3)
61826	Number of lines	1) 0 .. 255 2) 2 3)
61827	Number of characters (61825 * 61826)	1) 0 .. 255 2) 48 3)
61667	User interface LED (bit 0 .. 11, overlayed with registers 61488, 61489)	1) 0 .. 65535 2) 0 3)
62984	User interface baudrate (0 .. 15)	1) 0 .. 255 2) 6 (9600 Baud) 3)
	Network 1 Control	
61585	Network reaction time in ms	1) 0 .. 255 2) 0 3)
61586	Network instrcution execution time in ms	1) 0 .. 255 2) 0 3)

61588	Timeout time network 1	1) 0 .. 255 2) 3)
61589	Indirect network addressing	1) 0 .. 255 2) 0 3)
61828	50000er offset for inputs	1) 0 .. 65535 2) 100 3)
61829	50000er offset for outputs	1) 0 .. 65535 2) 200 3)
61830	50000er offset for flags	1) 0 .. 65535 2) 0 3)
61831	50000er offset for registers	1) 0 .. 65535 2) 0 3)
61955	Number checksum errors network receiving	1) -8388608 .. +8388607 2) 0 3)
62995	Network 1 number	1) -8388608 .. +8388607 2) Register 63241 3)
62996	Baud rate Network 1 0 150 1 300 2 600 3 1200 4 2400 5 4800 6 9600 default 7 19200 8 38400 9 57600 10 76800 11..14 115200	1) 0 .. 255 2) Register 63242 3)
63242	Baud rate Network EEPROM	
63241	Network number network 1 at switch-on (EEPROM)	1) 0 .. 255 2) last number 3)
Free Prog. Interface via LCD Connector instead of User Interface		
62155	Reception buffer level	1) 0 .. 255 2) 0 3)
62156	Transmission buffer level	1) 0 .. 255 2) 0 3)
62984	Baud rate 0 150 1 300 2 600	1) 0 .. 255 2) 6 3)

	3 1200 4 2400 5 4800 6 9600 default 7 19200 8 38400	
63002	Interface state Bit3=1 character lost during reception, parity error, stop bit error Bit4=1 receiving buffer overflow	1) 0 .. 256 2) State 3)
63003	Configuration Bit0 0=PRIM, 1=LCD Bit1 reserved Bit2 0=1, 1=2 stop bit Bit3 0=7, 1=8 bit/char Bit4,5 00=no parity 01=not allowed 10=odd 11=even Bit7,6 00=RS232 01=RS485 (4 wire) 10=RS422 11=RS485 (2 wire) Default: PRIM, RS232, 8N1,9600	1) 0 .. 255 2) 3)
63004	Transmission buffer	1) 0 .. 255 2) 3)
63005	Reception buffer - access removes character	1) 0 .. 255 2) 3)
63006	Reception buffer - access doed not remove character	1) 0 .. 255 2) 3)
Network 2 Control / Free Prog. Interface		
61499	Receiving buffer level	1) 0 .. 128 2) 0 3)
61502	Tansmission buffer level	1) 0 .. 128 2) 0 3)
61508	Interface state Bit0=1 one or more charecters lost during receiving Bit1=1 stop bit error Bit2=1 parity error Bit3=1 bit0 to 2 is or was not 0 Bit4=1 receiving buffer overflow	1) 0 .. 256 2) State 3)

	Bit5=1 transmission buffer overflow	
61511	Timeout time network 2	1) 0 .. 255 2) 3)
62989	Configuration Bit0 0=PRIM, 1=net2 Bit1 reserved Bit2 0=1, 1=2 stop bit Bit3 0=7, 1=8 bit/char Bit4,5 00=no parity 01=not allowed 10=odd 11=even Bit7,6 00=RS232 01=RS485 (4 wire) 10=RS422 11=RS485 (2 wire) Default: PRIM, RS232, 8N1,9600	1) 0 .. 255 2) 3)
62990	Baud rate 0 150 1 300 2 600 3 1200 4 2400 5 4800 6 9600 default 7 19200 8 38400 9 57600 10 76800 11..14 115200	1) 0 .. 255 2) 6 3)
62991	Network number network 2	1) 0 .. 255 2) 2 3)
62992	Transmission buffer	1) 0 .. 255 2) 0 3)
62993	Receiving buffer with character removing	1) 0 .. 255 2) 0 3)
62994	Receiving buffer without character removing	1) 0 .. 255 2) 0 3)
Programming Interface Control (PC)		
62983	Baud rate programming interface (0..15) 0 150 1 300 2 600 3 1200 4 2400	1) 0 .. 255 2) Register 63238 3)

	5 4800 6 9600 default 7 19200 8 38400 9 57600 10 76800 11..14 115200 initialized by 63238 at power-on	
63238	Baud rate pointer programming interface (EEPROM)	1) 0 .. 255 2) EEPROM 3)
User Programm State		
61440	Bit0=0 Program stopped Bit0=1 Program runs	1) 0 .. 255 2) 1 3)
61529	Switch position at power-on 0=Stop, 2=Run, 1=Load	1) 0 .. 255 2) Switch position power-on 3)
62976	Current switch position 0=Stop, 2=Run, 1=Load	1) 0 .. 255 2) Switch position 3)
Time Register		
61645	Number of time registers	1) 0 .. 255 2) Number time registers 3)
61708 .. 61739	Task time registers	1) 0 .. 65535 2) 0
61806	User time base in ms	1) 0 .. 65535 2) 3)
61848	Time base for START-TIMER, TIMER-END?	1) 0 .. 65535 2) 3)
Real Time Clock		
62912 .. 62919	RTC buffer - no transfer to RTC	1) 0 .. 255 2) 3)
62920 .. 62927	RTC direct, transfer of all registers	1) 0 .. 255 2) 3)
62912, 62920	Seconds	1) 0 .. 255 2) current time 3)
62913, 62921	Minutes	1) 0 .. 255 2) current time

		3)
62914, 62922	Hours	1) 0 .. 255 2) current time 3)
62915, 62923	24 hours format 0, 12 hours format: 0=am, 128=pm	1) 0 .. 255 2) last setting 3)
62916, 62924	Day of week, 1=Sunday	1) 0 .. 255 2) last setting 3)
62917, 62925	Day	1) 0 .. 255 2) current date 3)
62918, 62926	Month	1) 0 .. 255 2) current date 3)
62919, 62927	Year	1) 0 .. 255 2) current date 3)
63240	Configuration (EEPROM) Bit0=1 Summer-/Winter time activated Bit1=0 am/pm format	1) 0 .. 255 2) 3)
Floating Point Registers		
62208 .. 62463	Value range: -10^{15} to $+10^{15}$	1) -10^{15} bis $+10^{15}$ 2) last value 3)
Controller Module Access Values valid if 2104/2105=1		
62150	Controller module access error: slot number minus 1	1) 0 .. 255 2) 0 3)
62151	Axis module access error: axis number minus 1	1) 0 .. 255 2) 0 3)
62152	Controller module access error: register number	1) 0 .. 65535 2) 0 3)
General Registers		
61684	Slot and modules bit code	1) 0 .. 255 2) Slot state 3)
62985	Single channel counter at INPUT2	1) -8388608 .. +8388607 2) 0 3)

24 Combined Inputs		
62592	Inputs 101 .. 124	1) -8388608 .. +8388607
62593	Inputs 109 .. 132	2) Input state
62594	Inputs 117 .. 140	3)
62595	Inputs 125 .. 148	
62596	Inputs 133 .. 156	
62597	Inputs 141 .. 164	
62598	Inputs 149 .. 164	
62599	Inputs 157 .. 164	
62600	Inputs 201 .. 224	
62601	Inputs 209 .. 232	
62602	Inputs 217 .. 240	
62603	Inputs 225 ..248	
62604	Inputs 233 .. 256	
62605	Inputs 241 .. 264	
62606	Inputs 249 .. 264	
62607	Inputs 257 .. 264	
62608	Inputs 301 .. 324	
62609	Inputs 309 .. 332	
62610	Inputs 317 .. 340	
62611	Inputs 325 .. 348	
62612	Inputs 333 .. 356	
62613	Inputs 341 .. 364	
62614	Inputs 349 .. 364	
62615	Inputs 357 .. 364	
62616	Inputs 401 .. 424	
62617	Inputs 409 .. 432	
62618	Inputs 417 .. 440	
62619	Inputs 425 .. 448	
62620	Inputs 433 .. 456	
62621	Inputs 441 .. 464	
62622	Inputs 449 .. 464	
62623	Inputs 457 .. 464	
62624	Inputs 501 .. 524	
62625	Inputs 509 .. 532	
62626	Inputs 517 .. 540	
62627	Inputs 525 .. 548	
62628	Inputs 533 .. 556	
62629	Inputs 541 .. 564	
62630	Inputs 549 .. 564	
62631	Inputs 557 .. 564	
62632	Inputs 601 .. 624	
62633	Inputs 609 .. 632	
62634	Inputs 617 .. 640	
62635	Inputs 625 .. 648	
62636	Inputs 633 .. 656	
62637	Inputs 641 .. 664	
62638	Inputs 649 .. 664	
62639	Inputs 657 .. 664	
62640	Inputs 701 .. 724	
62641	Inputs 709 .. 732	
62642	Inputs 717 .. 740	

62643	Inputs 725 .. 748	
62644	Inputs 733 .. 756	
62645	Inputs 741 .. 764	
62646	Inputs 749 .. 764	
62647	Inputs 757 .. 764	
62648	Inputs 801 .. 824	
62649	Inputs 809 .. 832	
62650	Inputs 817 .. 840	
62651	Inputs 825 .. 848	
62652	Inputs 833 .. 856	
62653	Inputs 841 .. 864	
62654	Inputs 849 .. 864	
62655	Inputs 857 .. 864	
16 Combined Inputs		
62528	Inputs 101 .. 116	1) 0 .. 65535
62529	Inputs 109 .. 124	2) Input state
62530	Inputs 117 .. 132	3)
62531	Inputs 125 .. 140	
62532	Inputs 133 .. 148	
62533	Inputs 141 .. 156	
62534	Inputs 149 .. 164	
62535	Inputs 157 .. 164	
62536	Inputs 201 .. 216	
62537	Inputs 209 .. 224	
62538	Inputs 217 .. 232	
62539	Inputs 225 .. 240	
62540	Inputs 233 .. 248	
62541	Inputs 241 .. 256	
62542	Inputs 249 .. 264	
62543	Inputs 257 .. 264	
62544	Inputs 301 .. 316	
62545	Inputs 309 .. 324	
62546	Inputs 317 .. 332	
62547	Inputs 325 .. 340	
62548	Inputs 333 .. 348	
62549	Inputs 341 .. 356	
62550	Inputs 349 .. 364	
62551	Inputs 357 .. 364	
62552	Inputs 401 .. 416	
62553	Inputs 409 .. 424	
62554	Inputs 417 .. 432	
62555	Inputs 425 .. 440	
62556	Inputs 433 .. 448	
62557	Inputs 441 .. 456	
62558	Inputs 449 .. 464	
62559	Inputs 457 .. 464	
62560	Inputs 501 .. 516	
62561	Inputs 509 .. 524	
62562	Inputs 517 .. 532	
62563	Inputs 525 .. 540	

62564	Inputs 533 .. 548	
62565	Inputs 541 .. 556	
62566	Inputs 549 .. 564	
62567	Inputs 557 .. 564	
62568	Inputs 601 .. 616	
62569	Inputs 609 .. 624	
62570	Inputs 617 .. 632	
62571	Inputs 625 .. 640	
62572	Inputs 633 .. 648	
62573	Inputs 641 .. 656	
62574	Inputs 649 .. 664	
62575	Inputs 657 .. 664	
62576	Inputs 701 .. 716	
62577	Inputs 709 .. 724	
62578	Inputs 717 .. 732	
62579	Inputs 725 .. 740	
62580	Inputs 733 .. 748	
62581	Inputs 741 .. 756	
62582	Inputs 749 .. 764	
62583	Inputs 757 .. 764	
62584	Inputs 801 .. 816	
62585	Inputs 809 .. 824	
62586	Inputs 817 .. 832	
62587	Inputs 825 .. 840	
62588	Inputs 833 .. 848	
62589	Inputs 841 .. 856	
62590	Inputs 849 .. 864	
62591	Inputs 857 .. 864	
8 Combined Inputs		
62464	Inputs 101 .. 108	1) 0 .. 255
62465	Inputs 109 .. 116	2) Input state
62466	Inputs 117 .. 124	3)
62467	Inputs 125 .. 132	
62468	Inputs 133 .. 140	
62469	Inputs 141 .. 148	
62470	Inputs 149 .. 156	
62471	Inputs 157 .. 164	
62472	Inputs 201 .. 208	
62473	Inputs 209 .. 216	
62474	Inputs 217 .. 224	
62475	Inputs 225 .. 232	
62476	Inputs 233 .. 240	
62477	Inputs 241 .. 248	
62478	Inputs 249 .. 256	
62479	Inputs 257 .. 264	
62480	Inputs 301 .. 308	
62481	Inputs 309 .. 316	
62482	Inputs 317 .. 324	
62483	Inputs 325 .. 332	
62484	Inputs 333 .. 340	

62485	Inputs 341 .. 348	
62486	Inputs 349 .. 356	
62487	Inputs 357 .. 364	
62488	Inputs 401 .. 408	
62489	Inputs 409 .. 416	
62490	Inputs 417 .. 424	
62491	Inputs 425 .. 432	
62492	Inputs 433 .. 440	
62493	Inputs 441 .. 448	
62494	Inputs 449 .. 456	
62495	Inputs 457 .. 464	
62496	Inputs 501 .. 508	
62497	Inputs 509 .. 516	
62498	Inputs 517 .. 524	
62499	Inputs 525 .. 532	
62500	Inputs 533 .. 540	
62501	Inputs 541 .. 548	
62502	Inputs 549 .. 556	
62503	Inputs 557 .. 564	
62504	Inputs 601 .. 608	
62505	Inputs 609 .. 616	
62506	Inputs 617 .. 624	
62507	Inputs 625 .. 632	
62508	Inputs 633 .. 640	
62509	Inputs 641 .. 648	
62510	Inputs 649 .. 656	
62511	Inputs 657 .. 664	
62512	Inputs 701 .. 708	
62513	Inputs 709 .. 716	
62514	Inputs 717 .. 724	
62515	Inputs 725 .. 732	
62516	Inputs 733 .. 740	
62517	Inputs 741 .. 748	
62518	Inputs 749 .. 756	
62519	Inputs 757 .. 764	
62520	Inputs 801 .. 808	
62521	Inputs 809 .. 816	
62522	Inputs 817 .. 824	
62523	Inputs 825 .. 832	
62524	Inputs 833 .. 840	
62525	Inputs 841 .. 848	
62526	Inputs 849 .. 856	
62527	Inputs 857 .. 864	
24 Combined Outputs		
62848	Outputs 101 .. 124	1) -8388608 .. +8388607
62849	Outputs 109 .. 132	2) Output state
62850	Outputs 117 .. 140	3)
62851	Outputs 125 .. 148	
62852	Outputs 133 .. 156	
62853	Outputs 141 .. 164	

62854	Outputs 149 .. 164	
62855	Outputs 157 .. 164	
62856	Outputs 201 .. 224	
62857	Outputs 209 .. 232	
62858	Outputs 217 .. 240	
62859	Outputs 225 ..248	
62860	Outputs 233 .. 256	
62861	Outputs 241 .. 264	
62862	Outputs 249 .. 264	
62863	Outputs 257 .. 264	
62864	Outputs 301 .. 324	
62865	Outputs 309 .. 332	
62866	Outputs 317 .. 340	
62867	Outputs 325 .. 348	
62868	Outputs 333 .. 356	
62869	Outputs 341 .. 364	
62870	Outputs 349 .. 364	
62871	Outputs 357 .. 364	
62872	Outputs 401 .. 424	
62873	Outputs 409 .. 432	
62874	Outputs 417 .. 440	
62875	Outputs 425 .. 448	
62876	Outputs 433 .. 456	
62877	Outputs 441 .. 464	
62878	Outputs 449 .. 464	
62879	Outputs 457 .. 464	
62880	Outputs 501 .. 524	
62881	Outputs 509 .. 532	
62882	Outputs 517 .. 540	
62883	Outputs 525 .. 548	
62884	Outputs 533 .. 556	
62885	Outputs 541 .. 564	
62886	Outputs 549 .. 564	
62887	Outputs 557 .. 564	
62888	Outputs 601 .. 624	
62889	Outputs 609 .. 632	
62890	Outputs 617 .. 640	
62891	Outputs 625 .. 648	
62892	Outputs 633 .. 656	
62893	Outputs 641 .. 664	
62894	Outputs 649 .. 664	
62895	Outputs 657 .. 664	
62896	Outputs 701 .. 724	
62897	Outputs 709 .. 732	
62898	Outputs 717 .. 740	
62899	Outputs 725 .. 748	
62900	Outputs 733 .. 756	
62901	Outputs 741 .. 764	
62902	Outputs 749 .. 764	
62903	Outputs 757 .. 764	
62904	Outputs 801 .. 824	
62905	Outputs 809 .. 832	
62906	Outputs 817 .. 840	

62907	Outputs 825 .. 848	
62908	Outputs 833 .. 856	
62909	Outputs 841 .. 864	
62910	Outputs 849 .. 864	
62911	Outputs 857 .. 864	
16 Combined Outputs		
62784	Outputs 101 .. 116	1) 0 .. 65535
62785	Outputs 109 .. 124	2) Output state
62786	Outputs 117 .. 132	3)
62787	Outputs 125 .. 140	
62788	Outputs 133 .. 148	
62789	Outputs 141 .. 156	
62790	Outputs 149 .. 164	
62791	Outputs 157 .. 164	
62792	Outputs 201 .. 216	
62793	Outputs 209 .. 224	
62794	Outputs 217 .. 232	
62795	Outputs 225 .. 240	
62796	Outputs 233 .. 248	
62797	Outputs 241 .. 256	
62798	Outputs 249 .. 264	
62799	Outputs 257 .. 264	
62800	Outputs 301 .. 316	
62801	Outputs 309 .. 324	
62802	Outputs 317 .. 332	
62803	Outputs 325 .. 340	
62804	Outputs 333 .. 348	
62805	Outputs 341 .. 356	
62806	Outputs 349 .. 364	
62807	Outputs 357 .. 364	
62808	Outputs 401 .. 416	
62809	Outputs 409 .. 424	
62810	Outputs 417 .. 432	
62811	Outputs 425 .. 440	
62812	Outputs 433 .. 448	
62813	Outputs 441 .. 456	
62814	Outputs 449 .. 464	
62815	Outputs 457 .. 464	
62816	Outputs 501 .. 516	
62817	Outputs 509 .. 524	
62818	Outputs 517 .. 532	
62819	Outputs 525 .. 540	
62820	Outputs 533 .. 548	
62821	Outputs 541 .. 556	
62822	Outputs 549 .. 564	
62823	Outputs 557 .. 564	
62824	Outputs 601 .. 616	
62825	Outputs 609 .. 624	
62826	Outputs 617 .. 632	
62827	Outputs 625 .. 640	

62828	Outputs 633 .. 648	
62829	Outputs 641 .. 656	
62830	Outputs 649 .. 664	
62831	Outputs 657 .. 664	
62832	Outputs 701 .. 716	
62833	Outputs 709 .. 724	
62834	Outputs 717 .. 732	
62835	Outputs 725 .. 740	
62836	Outputs 733 .. 748	
62837	Outputs 741 .. 756	
62838	Outputs 749 .. 764	
62839	Outputs 757 .. 764	
62840	Outputs 801 .. 816	
62841	Outputs 809 .. 824	
62842	Outputs 817 .. 832	
62843	Outputs 825 .. 840	
62844	Outputs 833 .. 848	
62845	Outputs 841 .. 856	
62846	Outputs 849 .. 864	
62847	Outputs 857 .. 864	
8 Combined Outputs		
62720	Outputs 101 .. 108	1) 0 .. 255
62721	Outputs 109 .. 116	2) Output state
62722	Outputs 117 .. 124	3)
62723	Outputs 125 .. 132	
62724	Outputs 133 .. 140	
62725	Outputs 141 .. 148	
62726	Outputs 149 .. 156	
62727	Outputs 157 .. 164	
62728	Outputs 201 .. 208	
62729	Outputs 209 .. 216	
62730	Outputs 217 .. 224	
62731	Outputs 225 .. 232	
62732	Outputs 233 .. 240	
62733	Outputs 241 .. 248	
62734	Outputs 249 .. 256	
62735	Outputs 257 .. 264	
62736	Outputs 301 .. 308	
62737	Outputs 309 .. 316	
62738	Outputs 317 .. 324	
62739	Outputs 325 .. 332	
62740	Outputs 333 .. 340	
62741	Outputs 341 .. 348	
62742	Outputs 349 .. 356	
62743	Outputs 357 .. 364	
62744	Outputs 401 .. 408	
62745	Outputs 409 .. 416	
62746	Outputs 417 .. 424	
62747	Outputs 425 .. 432	
62748	Outputs 433 .. 440	

62749	Outputs441 .. 448	
62750	Outputs449 .. 456	
62751	Outputs457 .. 464	
62752	Outputs501 .. 508	
62753	Outputs509 .. 516	
62754	Outputs517 .. 524	
62755	Outputs525 .. 532	
62756	Outputs533 .. 540	
62757	Outputs541 .. 548	
62758	Outputs549 .. 556	
62759	Outputs557 .. 564	
62760	Outputs601 .. 608	
62761	Outputs609 .. 616	
62762	Outputs617 .. 624	
62763	Outputs625 .. 632	
62764	Outputs633 .. 640	
62765	Outputs641 .. 648	
62766	Outputs649 .. 656	
62767	Outputs657 .. 664	
62768	Outputs701 .. 708	
62769	Outputs709 .. 716	
62770	Outputs717 .. 724	
62771	Outputs725 .. 732	
62772	Outputs733 .. 740	
62773	Outputs741 .. 748	
62774	Outputs749 .. 756	
62775	Outputs757 .. 764	
62776	Outputs801 .. 808	
62777	Outputs809 .. 816	
62778	Outputs817 .. 824	
62779	Outputs825 .. 832	
62780	Outputs833 .. 840	
62781	Outputs841 .. 848	
62782	Outputs849 .. 856	
62783	Outputs857 .. 864	

Register - Flag Overlaying

RegNumber	**FlagNumber**	
0	256 .. 279	
1	280 .. 303	
...		
74	2024 .. 2047	

7. CPU Module Functions

7.1 User Interfaces, User Guidance

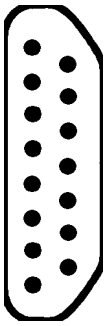
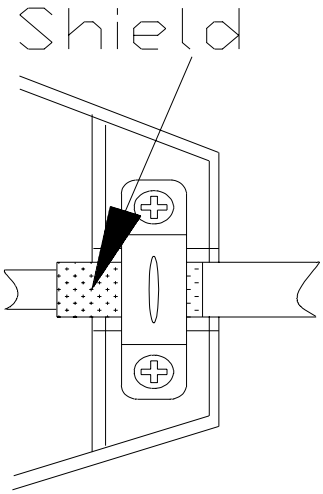
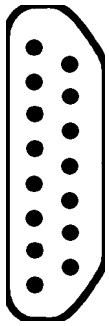
7.1.1 Overview, Technical Specifications

User Interfaces Overview

Type	Display	Keys	Remark	Interface Cable
LCD9	2 lines each 24 characters	12 F-keys (LED) Special function keys Decimal block		OpenColl EM-DK
LCD10	2 lines each 24 characters	12 F-keys (LED) Special function keys Decimal block	9mm char height illuminated	OpenColl EM-DK
LCD110	4 lines each 20 characters	12 F-keys (LED) Spezial function keys Decimal block	illuminated	RS422 DK-422
LCD12	2 lines each 16 characters	4 F-keys Special function keys Decimal block	designed for operation by manual operation systems	OpenColl EM-DK
LCD16	4 lines each 20 characters	5 F-keys (LED)	can be expanded by keyboard modules (NUM25) and handwheel modules (HR1)	RS422 DK-422

User Interface Overview				
Type	**Display**	**Keys**	**Remark**	**Interface Cable**
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 each 24 characters 1 lines each 16 characters 1 line each 8 characters (LED)	Cursor left Cursor right ENTER	5mm char height 8mm char height 12mm char height	RS422 DK-422
LCD25 LCD25L LED25	2 lines each 24 characters 1 lines each 16 characters 1 lines each 8 characters (LED)	5 F-keys (LED)	5mm char height illuminated 8mm char height illuminated 12mm char height	RS422 DK-422
LCD27	2 lines each 24 charecters	5 F-keys Cursor block Clear ENTER		RS422 DK-422
LCD34	2 lines each 24 characters	5 F-keys Special function keys Decimal block	illuminated	RS422 DK-422

7.1.2 Terminals

User Interface Cable (DK-422)			
PROCESS-PLC			User Interface
15 pin Sub-D-male 	RS422 max.Cable length: 400m  **Attach shield at both sides ! Use metallised housings !**		15 pol. Sub-D-male 
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 programming cable EM-PK can be obtained from JETTER.
If you make the cable yourself, the following minimum requirements must be considered:

Number of wires:	6
Diameter:	0,25mm²
Connection:	Sub-D, metallised
Shielding:	total, not paired

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

7.1.3 Programming of the User Interfaces: DISPLAY_TEXT, DISPLAY_REG, USER_INPUT

This chapter explains the instructions that are available for control of the user interfaces.

Instructions for user interface control:

DISPLAY_TEXT (2)

DISPLAY_REG

USER_INPUT

Text Display

The instruction

DISPLAY_TEXT [#<DeviceNo>, cp=<Cursorpos> "<Text>"]

displays text on the user interface display.

Meaning of the Parameters

Device number

The numbers 0 to 10 are valid parameters.

#0 to #4

a user interface is accessed

#8

Centronics printer interface

#9, #10

The free programmable (PRIM) interface is accessed.

Cursor Position

This parameter defines the position of the first character of the text to be displayed.

Cursor Positions of the User Interfaces

Type	Cursor position
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
LCD16 LCD110	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 8
LCD25	1. line: 1 to 24 2. line: 25 to 48

LCD25L LED25	1 to 16 1 to 8
LCD27	1. line: 1 to 24 2. line: 25 to 48
LCD34	1. line: 1 to 24 2. line: 25 to 48

By cursor position 0 the text will be attached after the last character

The **0** cursor position has a special meaning: If cursor position 0 is set, the latest text is attached to the text edited last. The cursor will appear at exactly the same position, where it had been after having carried out the last user interface instruction (if Reg 61648 = 0).

Text

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

_ clears display

"**_**" By this character, first, deleting the display, and then display of the given text is triggered, 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 this sign is to be displayed, the sign code for DELSCR can be changed in the special register 61462.

$ clears line from cursor position

"**$**" 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).
At the cursor position, 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 61648

If in register 61648 anything else but zero has been written, the register contents is interpreted as cursor position and the text **"ERROR"** written at this position.

Register 61683 selects between two alternative display texts

4) **DISPLAY_TEXT_2 [#0, cp=1, "Text 1"]**
"Text 2"

The content of register 61683 selects between display of **"Text 1"** or **"Text 2".**

Register 61683	Display Content
0	Text 1
1	Text 2

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 exactly the same function as the DISPLAY_TEXT instruction (see above). Additionally, a **register number** must be input here. Of course, this is the number of the register the contents of which is to be displayed. 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 displayed. If register 61453 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=121009]

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 121009 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. (Assumptions: The actual position of axis 21 is to have value 5400.

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

The dots are to represent the positions, which, after issuing the instructions, have still 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 using a user interface.

For the two 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 61649 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 to be assigned to. Here, a simple indirect register address is possible as well.

Note, that for USER_INPUT there are normally 8 characters available. This value (format of the user input), which has been stored in register 61455, can also be altered.

Example:

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

In order to realise optimum operator guidance, the USER_INPUT instruction is combined with the DISPLAY_TEXT instruction in most cases.
After issuing those two instructions, the "New position ?" instruction will be displayed at the left of the upper line, and after that, the input of a number is awaited. This number, which will be stored in register 100, will serve as a new nominal position for a positioning run.

Surveillance of the **USER_INPUT** instruction

Timeout time in register 61818

The user input can be surveilled by a timeout. Register 61818 defines a time in seconds within which the user input has to be completed. If user input is not completed by (<┘)within this time the user input is terminated and flag 2053 is set. The destination register of the user input is not modified.

Flag 2053 U terminated

Termination of the **USER_INPUT** instruction

Flag 2051 terminates U

Flag 2051 is active during execution of the USER-INPUT instruction. The user input is terminated if this flag is reset from an other task.

Flag 2053=0 destination register stores input

In this case flag 2053 defines if the entered value is transferred into the destination register (if already input) or if the content of the destination register remains unchanged.

Flag 2053=1 destination register ignores input

Free programmable Interface and **USER_INPUT**, and **DISPLAY_TEXT**

DISPLAY_TEXT, DISPLAY_REG

Output to free programmable interface with device number

#9, #10

The DISPLAY_TEXT and DISPLAY_REG instructions can address the free programmable interface. Therefor the #9, #10 parameters can be used.

Example:

```
DISPLAY_TEXT [#9, cp=1, "Text to FreeProgInterface"]
```

Note:

The control characters _ and $ for clearance of the display are not interpreted but directly output.

USER_INPUT

Input via free programmable interface with device number

#9, #10

The USER_INPUT instruction receives **numerical characters** via the interface and fit them together to number values. The reception of a **not numerical character** finishes the creation of the value. The value is assigned to the destination register.

Flag 2051 or timeout terminate the USER_INPUT instruction.

Example:

```
USER_INPUT [#9, cp=1, REG=100]
```

Note:

Numerical characters are

Integer register: 0 to 9 and - (sign)

Floating point: as above and . (point)

Each minus character toggles the sign like a pocket calculator (+/-)

7.1.4 User Interface Registers

User Interface Registers Overview		
User Interface Control (LCD-Display)		
61448	Display language, 0 = german, 1 = english	1) 0 .. 255 2) Depends on the user program 3)
61451	Field width for floating point register display	1) 0 .. 255 2) 8 3)
61452	Number of decimal places	1) 0 .. 255 2) 4 3)
61453	Field width for integer display	1) 0 .. 255 2) 1 3)
61454	Flush left number display Flag 2055=0 leading blanks Flag 2055=1 leading zeros	1) 0 .. 255 2) 0 3)
61455	Field width USER_INPUT	1) 0 .. 255 2) 8 3)
61461	'Delete line' character	1) 0 .. 255 2) 36 3)
61462	'Clear display' character	1) 0 .. 255 2) 95 3)
61472	Restrictions for monitor functions 0 = disable, 1 = enable (Flag 2096 .. 2103) Bit0=0 R, I/O key without register number input Bit0=1 R, I/O key with register number input Bit1=0 R, I/O key without flag number input Bit1=1 R, I/O key with flag number input Bit2=0 R, I/O key without output number input Bit2=1 R, I/O key with output	1) 0 .. 255 2) 0 3)

	number input Bit3=0 R, I/O key without input number input Bit3=1 R, I/O key with output number input Bit4=0 = key changes no register contents Bit4=1 = key changes register contents Bit5=0 = key changes no flags Bit5=1 = key changes flags Bit6=0 = key changes no outputs Bit6=1 = key changes outputs Bit7=0 = key does not access to inputs Bit7=1 = key does access to inputs	
61480 .. 61487	User interface keys (Special flags 2169 .. 2223)	1) 0 .. 255 2) Key status 3) See special registers
61488 .. 61489	LED user interfaces (Special flags)	1) 0 .. 255 2) 0 3) See special registers
61648	Indirect cursor position cp=0 for DISPLAY_TEXT und DISPLAY_REG	1) 0 .. 255 2) 0 3)
61649	Indirect cursor position cp=0 for USER_INPUT	1) 0 .. 255 2) 0 3)
61653	Display time for monitor functions (s)	1) 0 .. 65535 2) 3 3)
61683	Text selection for DISPLAY_TEXT_2 0 = text 1, 1 = text 2	1) 0 .. 255 2) 0 3)
61967	First register changeable by user interface - range 1	1) -8388608 .. +8388607 2) 0 3)
61968	Last register changeable by user interface - range 1	1) -8388608 .. +8388607 2) +8388607 3)
61969	First register changeable by user interface - range 2	1) -8388608 .. +8388607 2) 0 3)
61970	Last register changeable by user interface - range 2	1) -8388608 .. +8388607 2) +8388607 3)
61971	First register changeable by user interface - range 3	1) -8388608 .. +8388607 2) 0 3)
61972	Last register changeable by user interface - range 3	1) -8388608 .. +8388607 2) +8388607 3)
61973	First flag changeable by the	1) 0 .. 65535

	user interface	2) 0 3)
61974	Last flag changeable by the user interface	1) 0 .. 65535 2) 65535 3)
61818	Max. time for USER_INPUT default = 0 (disabled)	1) 0 .. 65535 2) 0 3)
61825	Number character per line	1) 0 .. 255 2) 24 3)
61826	Number of lines	1) 0 .. 255 2) 2 3)
61827	Number of characters (61825 * 61826)	1) 0 .. 255 2) 48 3)
61667	User interface LED (bit 0 .. 11, overlayed with registers 61488, 61489)	1) 0 .. 65535 2) 0 3)
62984	User interface baudrate (0 .. 15)	1) 0 .. 255 2) 6 (9600 Baud) 3)

Register 61827: User Interface Character Number

Function	Description
Read	Current number of display characters Value after reset: 48
Write	New number of characters
Value range	1 - 127

This register is initialized by the connected user interface.

Register 61825: Characters per Line

Function	Description
Read	Current number of characters per line Value after reset: 24
Write	New number of characters per line
Value range	1 - 127

This register is initialized by the connected user interface.

Register 61683: Text Selection for the DISPLAY_TEXT_2 Instruction	
Function	**Description**
Read	Current value for the display text of the DISPLAY_TEXT_2 instruction. Value 0: text 1 Value 1: text 2 Value after reset: 0
Write	New value for text selection: Value 0: text 1 Value 1: text 2
Value range	0 - 1

For bilingual text output

Two alternative texts can be displayed by the DISPLAY_TEXT_2 instruction. This allows bilingual user guidance. For instance text 1 for the customer text 2 for the service personal. The register selects between the two display texts.

Register 61453: Field Width for DISPLAY_REG	
Function	**Description**
Read	Current field width for DISPLAY_REG Value after reset: 1
Write	New field width
Value range	0 - 8

Number of reserved places for display

Number of places that have to be displayed. Up to 8 places can be reserved for register display.
If only 2 or 3 places are to be displayed on the user interface register 61453 reserves the actually necessary places. A lot of texts can be displayed this way.

Note:

The sign always occupies 1 place on the display. Register 61453 has to contain 7 if a 6 digit value with sign is to be displayed.

Register 61454: Flush Left Number Display	
Function	**Description**
Read	Defines sign and alignment of the number display Value after reset: 0
Write	New sign and alignment
Value range	0 - 3

Display format

SpecialReg "Flush left number display" 61454

This register defines if the content of a register is displayed flush left or flush right on the user interface. The value after reset is 0.

Following tables show various combinations of registers 61453 and 61454 and their effect on the display format. The cursor position always corresponds to the first character. The ***** (asterisk) represents in the following the display of a space character. The **+** character represents a positive sign, but in reality a space is displayed.

a) Special register **"Flush left number display"** = 0 (default after reset)

RegVal	1234	-345	7654321	-1234567
0	***+1234	****-345	+7654321	-1234567
1	+***1234	-****345	+7654321	-1234567
2	+**1234	-***345	+654321	-234567
3	+*1234	-**345	+54321	-34567
4	+1234	-*345	+4321	-4567
5	+234	-345	+321	-567
6	+34	-45	+21	-67
7	+4	-5	+1	-7
8	+	-	+	-

b) Special register **"Flush left number display"** = 1

RegVal	1234	-345	7654321	-1234567
0	+1234	-345	+7654321	-1234567
1	+	-	+	-
2	+1	-3	+7	-1
3	+12	-34	+76	-12
4	+123	-345	+765	-123
5	+1234	-345*	+7654	-1234
6	+1234*	-345**	+76543	-12345
7	+1234**	-345***	+765432	-123456
8	+1234***	-345****	+7654321	-1234567

c) Special register **"Flush left number display"** = 2 corresponds to point a) but no sign is displayed.

d) Special register **"Flush left number display"** = 3 corresponds to point b) and considers additionally flag 2055.

Special flag 2055: leading blanks / zeros

Flag 2055	Meaning
Flag 2055=0(default	leading blanks if flush right and Reg 61453≠0
Flag 2055=1	leading zeros if flush right and Reg 61453≠0

Register 61455: Field Width for USER_INPUT	
Function	**Description**
Read	Current field width for the USER_INPUT instruction Value after reset: 0
Write	New field width
Value range	1 - 8

Reserved number of places for value input

Defines the number of places for value input via the user interface. As a maximum 8 places can be reserved for an user input. If only 2 or 3 digit values are to be entered register 61455 allows to reserve only the actually needed places. This is necessary if a lot of texts and numbers have to be displayed on the user interface.

Note:

The sign always occupies 1 place on the display. Register 61453 has to contain 7 if a 6 digit value with sign is to be displayed.

Register 61648: Indirect Cursor Position for DISPLAY_TEXT, DISPLAY_REG	
Function	**Description**
Read	Current value for the indirect cursor position Value after reset: 0
Write	New indirect cursor position
Value range	0 - 127

If the cursor position parameter of the DISPLAY_TEXT, DISPLAY_REG instructions is 0 the cursor position defined in register 61648 is used. If the content of this register is also 0, the display text/value is annexed to the last output.

Register 61649: Indirect Cursor Position for USER_INPUT	
Function	**Description**
Read	Current indirect cursor position Value after reset: 0
Write	New indirect cursor position
Value range	0 - 127

If the cursor position parameter of the USER_INPUT instructions is 0 the cursor position defined in register 61649 is used. If the content of this register is also 0, the display value is annexed to the last output.

Register 61472: Restrictions for Monitor Functions	
Function	**Description**
Read	Current restrictions Value after reset: 255
Write	New restrictions, bitcoded
Value range	0 - 255

Locking of keyboard ranges for the machine operator

This register defines the keyboard functions that are available for the macine operator.
The register can be used to release keyboard functions that are locked for the service personal.

Register bit 61472	Function
Bit 0 = 1	Display of regisers by ´R´ key
Bit 0 = 0	´R´ key has no function
Bit 1 = 1	Display of flags (2 times ´R´)
Bit 1 = 0	No display of flags
Bit 2 = 1	Display of outputs by ´I/O´
Bit 2 = 0	No display of outputs
Bit 3 = 1	Display of inputs (2 times I/O´)
Bit 3 = 0	No display of inputs
Bit 4 = 1	Value assignment to registers by ´=´
Bit 4 = 0	No assignment to registers
Bit 5 = 1	Flag state change by ´=´
Bit 5 = 0	No change of flags
Bit 6 = 1	Output state change by ´=´
Bit 6 = 0	No output state change
Bit 7 = 1	Input display by ´=´
Bit 7 = 0	No input display

Register 61653: Display Time for Monitor Functions	
Function	**Description**
Read	Current switch over time from monitor - standard display in s Value after reset: 3 (s)
Write	New switch over time
Value range	0 - 65535

Switch over time monitor - standard display

Using the monitor mode the user can access to inputs, outputs, flags and registers via the user interface.
Register 61653 defines the time for switching from the monitor screen to the standard screen. The switch over takes place after the monitor input is confirmed by (ENTER). A value of 3 in register 61653 delays the switch over for 3 seconds.

Register 61448: Display Language	
Function	**Description**
Read	Current user interface dialogue language: **0**: German **1**: English Value after reset: 0
Write	New user interface dialogue language: **0**: German **1**: English
Value range	0 - 1

The register defines the dialogue language of the user interface. This dialogue language refers to the operating system functions of the user interface not to the display texts of the user program, for example the monitor function as register, input, output and flag access.

Note:

The register takes over the dialog language of SYMPAS during program transfer.

7.1.5 User Interface Flags

Flag 2112: User Interface Operation after each User Task	
Function	**Description**
Read	Current user interface priority: **0**: the user interface is served after execution of all user task **1**: the user interface is served after each user task Value after reset: 0
Write	Set for high, clear for low priority of the user interface

User interface priority

Definition of the user interface priority. The user interface is served in a background task. In most cases the user interface priority is lower than the priority of the user program. Therefore the user interface is served after execution of all user tasks. Normally this is sufficient since the delay times are about 1/100s and the user does not consider this as delay.
The priority of the user interface can be increased by setting of flag 2112 if a large amout of values are displayed especially on the 4 line user interfaces. Now the user interface is served after each user task (task1, UI, task2, UI, task3, UI, etc.).

Note:

Normally the user interface is operated with low priority. If considerable delays are recognised during user input, setting of flag 2112 increases the priority. In most cases complex display and input functions are needed during adjustment and manual operation of the machine. Thus there is the possibility to set the flag (high priority) during manual operation and clear the flag (low priority) for the automatic mode.

User Interface Control Keys and LED

User Interface LED Control

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

Special Function Keys			
<-	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
Numerical 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

User Interface Control			
2051	the USER_INPUT instruction is active		
2053	USER_INPUT terminated by timeout		
2112	User interface operated after each user task (high priority)		
User Interface Restrictions			

2052	Input by user interface locked		
2054	ENTER key does not switches to monitor mode		
2096	No register display	0 = locked	
2097	No flag display	0 = locked	
2098	No output display	0 = locked	
2099	No input display	0 = locked	
2100	No change of registers	0 = locked	
2101	No change of flags	0 = locked	
2102	No change of outputs	0 = locked	
2103	No continuous display of inputs		

7.2 Network Operation

7.2.1 JETWay-H: JETTER Data Highway

Access to up to 126 network participants by one host via JETWay-H

Several controllers of the PROCESS-PLC family can be accessed from one host via data highway JETWay-H. This means:

- Process monitoring
- Programming
- Data transfer
- Operation data aquisition
- Maintenance, of each controller directly

Remote maintenance is possible if a modem is used.

The maximum number of network participants is 126.

Note:

For terminal description and parametering of the JETWay-H see Chapter *6.3.2.2 PC Programming Interface (JETWay-H)*

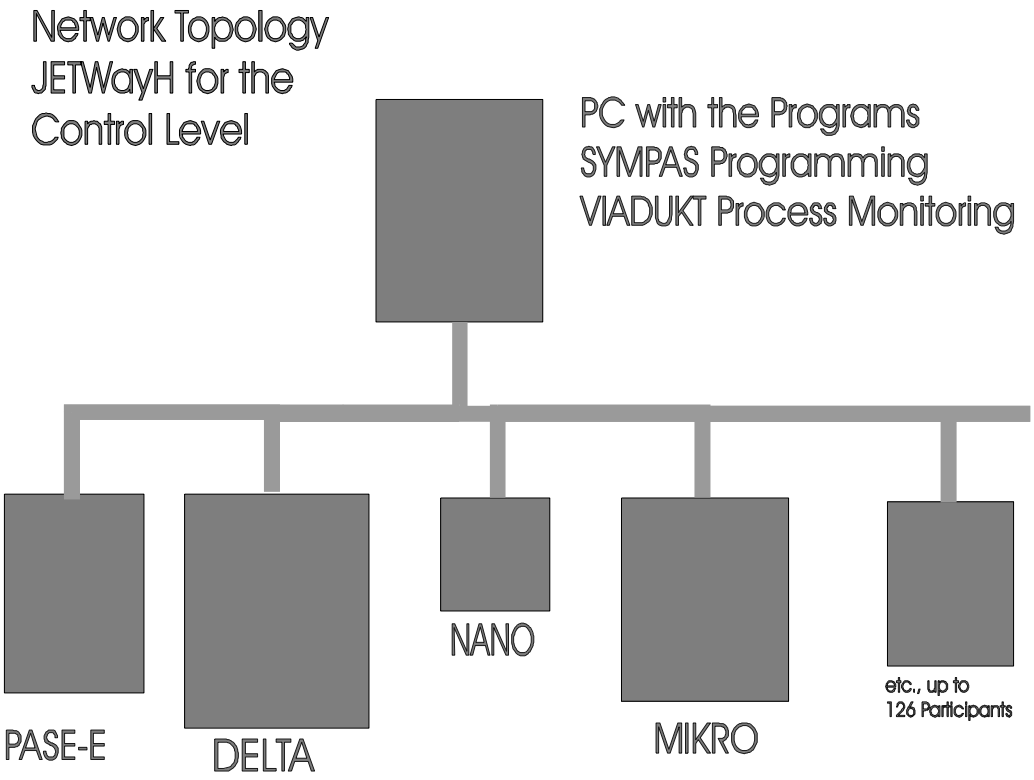


Figure 16: JETWay-H for the central controller

7.2.2 JETWay-R: The Process Level

The JETWay-R network provides two functions:

1. The hierarchic connection of PROCESS-PLC in networks
2. Connection of decentralized peripheral devices like remote I/O and valve blocks

The maximum number of participants of each level is 126. JETWay is a mono master network, that means there are 1 master and 125 slaves.

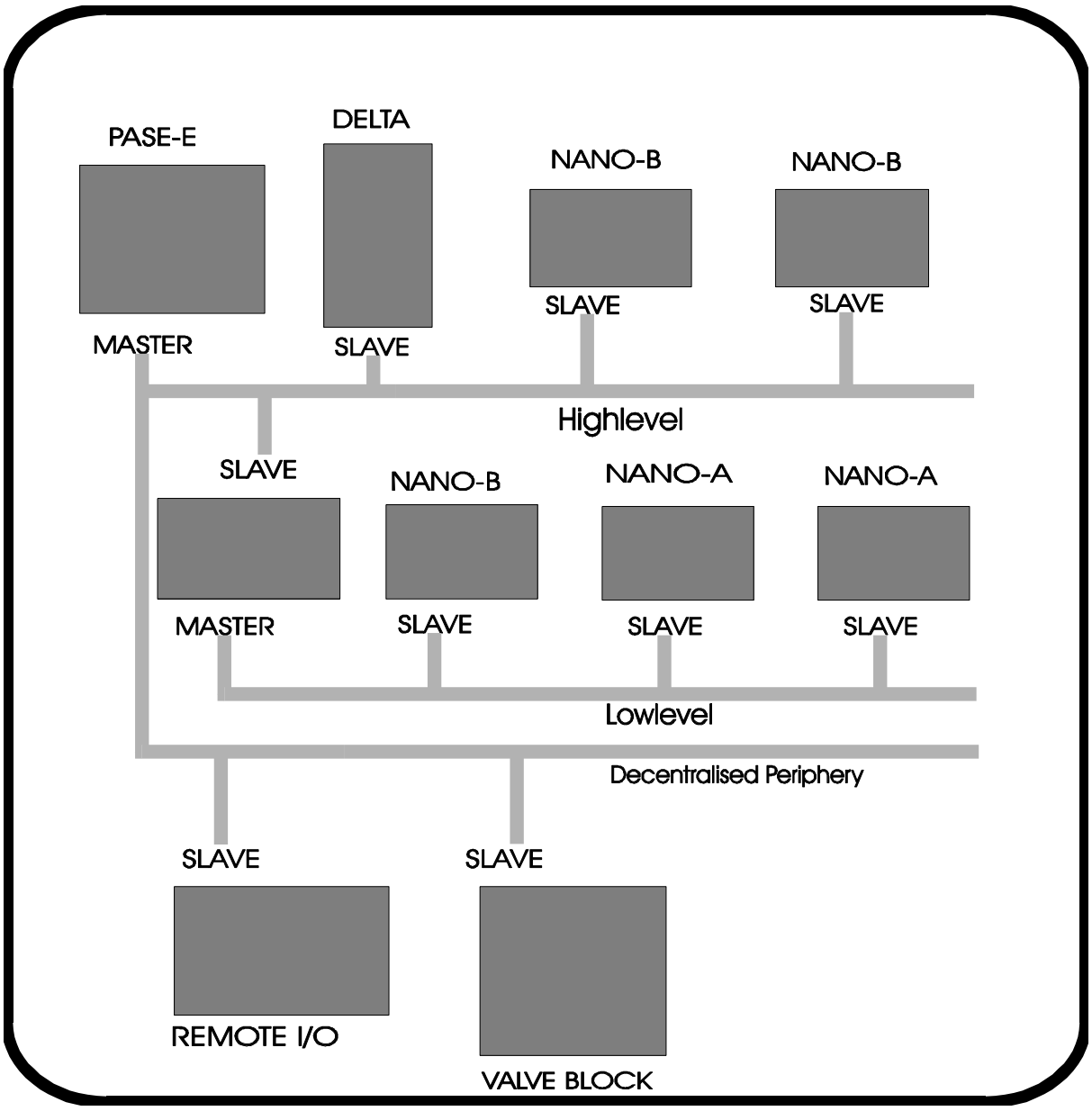
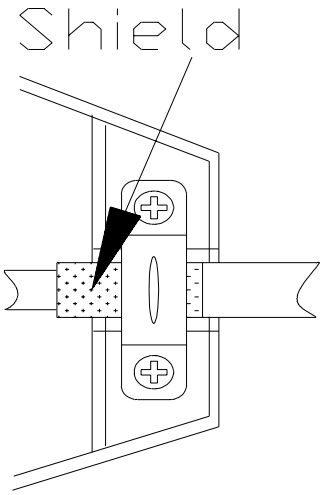


Figure 8: JETWay-R for the process level

7.2.3 Terminals

The JETWay-R is used to connect PROCESS-PLC and/or PROCESS-PLC and remote I/O, valve blocks, etc.

JETWay-R Cable		
Connection on DELTA side	**Shielding**	**Specification max. Length**
9 pin Sub-D-male **NET1** or 9 pin Sub-D-male **NET2**	 **Attach shield at both sides !** **Use metallised housings !**	RS485 max. cable length: 400m
Pin	**Signal**	**Remark**
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:	0,25mm²
Connection:	Sub-D, metallised
Shielding:	total, not paired

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

7.2.4 Network Access by 50000er Numbers

Note:

- The PROCESS-PLC DELTA can be master or slave in the JETWay-R.
- Both networks can be used simultaneously. Only one can be master at the same time.

7.2.4.1 Addressing the Registers

Addressing registers of a controller by the master control only differs from an internal REGISTER_LOAD command 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

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

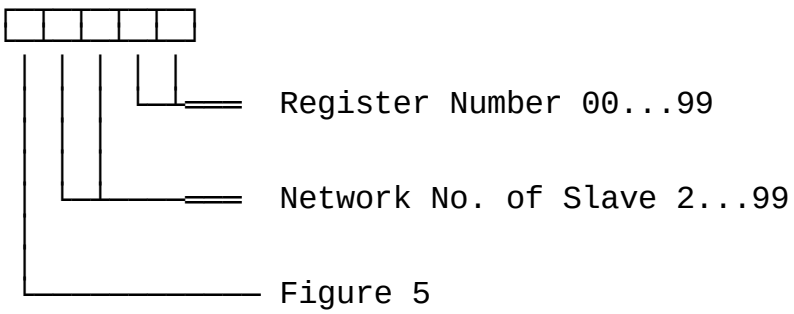


Figure 5

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 which number is greater than 99 is to be addressed, a numeric offset value is to be written into register 61831 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 input offset register is in the slave control

Register 61831:

Number offset for register; the register is on the slave control.

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 slave control, which is accessed by the master control.

The command

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

in the master controller program, plus value 100 written into register 61831 of the slave controller of network number 32 serves for actually addressing register 162 of the slave controller.

7.2.4.2 Addressing of Inputs, Outputs, and Flags

Addressing of Inputs

Access to inputs of the slave controller by the master control only differs from an internal master input command 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:

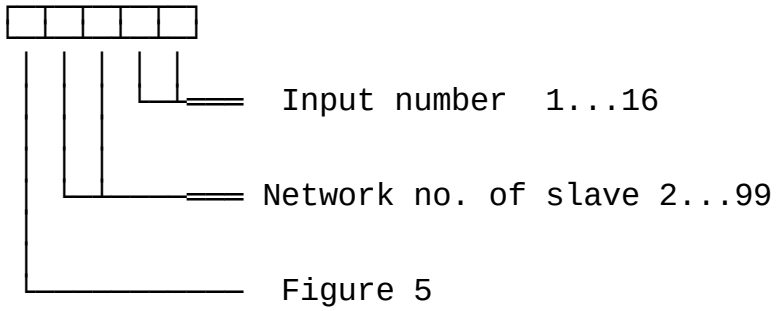


Figure 5

The input offset register is in the slave control

Register 61828:

Number offset for input; the register is on the slave controller.

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 slave controller, which is accessed by the master controler.

Value after Reset: 100

Example:

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

```
INPUT 50412
```

Note:

The numbering of the inputs in the different slots has to be considered for definition of the register number offset content.

Reg 61828 = slot number x 100

Addressing the Outputs

Access to outputs of the slave control by the master control only differs from an internal master output command 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:

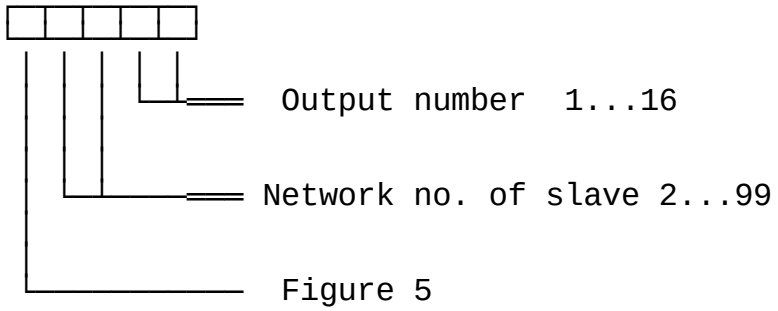


Figure 5

The output offset register is in the slave control

Register 61829:

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

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

Value after Reset: 100

Example:

Output 113 in the slave control with network number 4 is addressed by the master control by

```
OUTPUT 50413.
```

Note:

The numbering of the outputs in the different slots has to be considered for definition of the register number offset content.

Reg 61829 = slot number x 100

Addressing the Flags

Access to flags of the slave control by the master control only differs from an internal master flag command 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:

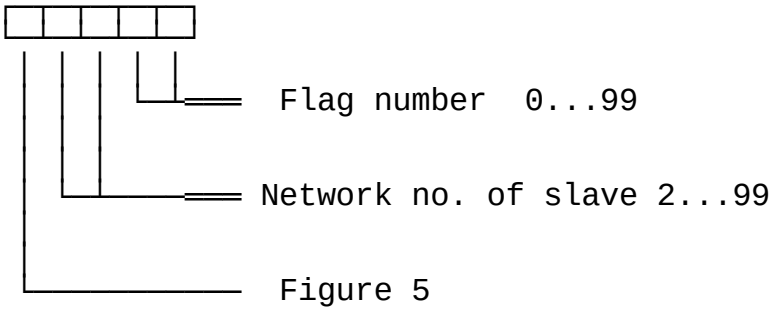


Figure 5

The flag offset register is in the slave control

Register 61830:

Number offset for the flag; the register is on the slave controller.

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

Value after Reset: 0

Example:

Flag 154 in the slave control with network number 12 is addressed by the master control by

```
FLAG 51254.
```

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

7.2.5 Network Access by **N-SEND REGISTER** and **N-GET REGISTER**

Note:

These register numbers are not influenced by the number offset in register 61831.

Note:

The PROCESS-PLC DELTA can be operated as master or slave in JETWay-R .

The N-SEND-REGISTER Instruction

By the following instruction, registers can be written into slave controllers by the master controller:

```
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 controller 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 controller is transferred into. This register is on the slave controller of the slave number PASE- no.

Example:

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

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

The N-GET Register Instruction

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

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

PASE NO.

PASE NO. is the network number of the slave controller which is to be addressed via network.

Source Reg

The source reg is the number of the register in the slave controller, which is to be written into the master controller.

Destination Reg

The destination register is the number of the master control register into which the value of the slave register is to be written.

Example:

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

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

7.2.6 Network Registers

Overview Network Registers		
Network 1 Control		
61585	Network reaction time in ms	1) 0 .. 255 2) 0 3)
61586	Network instrcution execution time in ms	1) 0 .. 255 2) 0 3)
61588	Timeout time network 1	1) 0 .. 255 2) 3)
61589	Indirect network addressing	1) 0 .. 255 2) 0 3)
61828	50000er offset for inputs	1) 0 .. 65535 2) 100 3)
61829	50000er offset for outputs	1) 0 .. 65535 2) 200 3)
61830	50000er offset for flags	1) 0 .. 65535 2) 0 3)
61831	50000er offset for registers	1) 0 .. 65535 2) 0 3)
61955	Number checksum errors network receiving	1) -8388608 .. +8388607 2) 0 3)
62995	Network 1 number	1) -8388608 .. +8388607 2) Register 63241 3)
62996	Baudrate Network 1 0 150 1 300 2 600 3 1200 4 2400 5 4800 6 9600 7 19200 8 38400	1) 0 .. 255 2) Register 63242 3)

	9 57600 10 76800 11..14 115200 11=default	
63242	Baud rate Network EEPROM	
63241	Network number network 1 at switch-on (EEPROM)	1) 0 .. 255 2) last number 3)
Network 2 Control / Free Prog. Interface		
61499	Receiving buffer level	1) 0 .. 128 2) 0 3)
61502	Tansmission buffer level	1) 0 .. 128 2) 0 3)
61508	Interface state Bit0=1 one or more charecters lost during receiving Bit1=1 stop bit error Bit2=1 parity error Bit3=1 bit0 to 2 is or was not 0 Bit4=1 receiving buffer overflow Bit5=1 transmission buffer overflow	
61511	Timeout time network 2	1) 0 .. 255 2) 3)
62989	Configuration Bit0 0=PRIM, 1=net2 Bit1 reserved Bit2 0=1, 1=2 stop bit Bit3 0=7, 1=8 bit/char Bit4,5 00=no parity 01=not allowed 10=odd 11=even Bit7,6 00=RS232 01=RS485 (4 wire) 10=RS422 11=RS485 (2 wire) Default: PRIM, RS232, 8N1,9600	1) 0 .. 255 2) 3)
62990	Baud rate 0 150 1 300 2 600 3 1200 4 2400 5 4800 6 9600 default 7 19200 8 38400	1) 0 .. 255 2) 6 3)

	9 57600 10 76800 11..14 115200	
62991	Network number network 2	1) 0 .. 255 2) 2 3)
62992	Transmission buffer	1) 0 .. 255 2) 0 3)
62993	Receiving buffer with character removing	1) 0 .. 255 2) 0 3)
62994	Receiving buffer without character removing	1) 0 .. 255 2) 0 3)

Each PROCESS-PLC system provides at least 1 interface for the JETWay network from JETTER. The registers menitioned here specifiy the network number and the transfer parameters.

Register 61585: Network Reaction Time

Function	Description
Read	Reaction time in ms Value after reset: last time
Write	prohibited
Value range	0 .. 255

The time starts with the start of the network instruction and stops after response of the slave controller.

Register 61586: Network Instruction

Execution Time	
Function	**Description**
Read	execution time in ms Value after reset: last time
Write	prohibited
Value range	0 .. 255

Contains the execution time of the last executed network instruction. This execution time consists of the network reaction time plus the time that is necessary to execute the response of the slave controller.

Register61588: Timeout Time Network 1	
Function	**Description**
Read	Timeout time in ms Value after reset: 250
Write	Set new Timeout
Value range	0 .. 255

Register 61589: Indirect Network Addressing	
Function	**Description**
Read	Current indirect network number Value after reset: 0
Write	New indirect network number
Value range	0 .. 255

Input offset

Register 61828: Input Offset	
Function	**Description**
Read	Current input offset Value after reset 100
Write	New inpu offset
Value range	0 - 65535

This value is added to the input number of a network access by 50000er numbers. See *Chapter 7.2.4.2* .

Output offset

Register 61829: Output Offset	
Function	**Description**
Read	Current output offset Value after eset: 200
Write	New output offset
Value range	0 - 65535

This value is added to the output number of a network access by 50000er numbers. See *Chapter 7.2.4.2* .

Flag offset

Register 61830: Flag Offset	
Function	**Description**
Read	Current flag offset Value after reset 0
Write	New flag offset

Value range	0 - 65535

This value is added to the flag number of a network access by 50000er numbers. See *Chapter 7.2.4.2* .

Register offset

Register 61831: Register Offset	
Function	**Description**
Read	Current register offset Value after reset: 0
Write	New register offset
Value range	0 .. 65535

This value is added to the input number of a network access by 50000er numbers. See *Chapter 7.2.4.1 .*

Register 62995: Network Number Network 1	
Function	**Description**
Read	Current number of the JETWay paticipant Value after reset: see Reg 63241
Write	New JETWay network number: **0**: not active **1**: master **2 - 127**: possible slave numbers
Value range	0 .. 127

Register 62996: Baud Rate Network 1	
Function	**Description**
Read	Current value for the JETWay-R baud rate Value after reset: Reg 63242
Write	New JETWay-R baud rate **0** = 150, **2** = 300, **3** = 600, **4** = 1200, **5** = 2400, **6** = 4800, **7** = 19200, **8** = 38400, **9** = 57600, **10** = 76800, **11** = 115200
Value range	1.. 255

Register 63241: Network Number Network 1 at Power-On (EEPROM)	
Function	**Description**
Read	Network number at power-on Value after reset: last number
Write	New network number at power-on
Value range	0 .. 255

See register 62995 network number network 1. Register 62995 is initialized with the content of register 63141 during power-up if register 62995 contains a number in the range of 2 .. 126 (only slave number are valid). If register 63241 contains an invalid network number register 62995 is initialized with 2 (slave 2).

Register 63242: Baud Rate Network 1 at Power-On (EEPROM)

Function	Description
Read	Baud rate at power-on Value after reset: last number
Write	New baud rate at power-on
Value range	0 .. 255

See register 62996 baud rate network 1.

Register61511: Timeout Time Network 2

Function	Description
Read	Timeout time in ms Value after reset: 250
Write	Set timeout time
Value range	0 .. 255

Register 62990: Baud Rate Network 2

Function	Description
Read	Current baud rate JETWay-R Value after reset: 6 (9600)
Write	new baud rate JETWay-R **0** = 150, **2** = 300, **3** = 600, **4** = 1200, **5** = 2400, **6** = 4800, **7** =

	19200, **8** = 38400, **9** = 57600, **10** = 76800, **11** = 115200
Value range	1 .. 255

Register 62991: Network Number Network 2

Function	Description
Read	Current number of JETWay participant Value after reset: 2
Write	New network number: **0**: not active **1**: master **2 - 127**: possible slave numbers
Value range	0 .. 255

Register 62989: Configuration NET2 - Free Programmable Interface

Function	Description
Read	Current configuration Value after reset: 8
Write	New configuration: **Bit0** 0=free prog, 1=Net 2 **Bit1** reserved **Bit2** 0=1, 1=2 stop bit **Bit3** 0=7, 1=8 bit / char **Bit4,5** 00=no parity 01=prohibited 10=odd

	11 = even **Bit7,6** 00 = RS232 01 = RS485 (4 wire) 10 = RS422 11 = RS485 (2 wire) **Default: PRIM, RS232, 8N1,9600**
Value range	0 .. 255

Note:

If bit 0 = 1 (NET2 selected) all configurations are ignored and not applied to the JETWay.

7.3 Free Programmable Interfaces

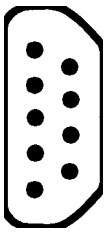
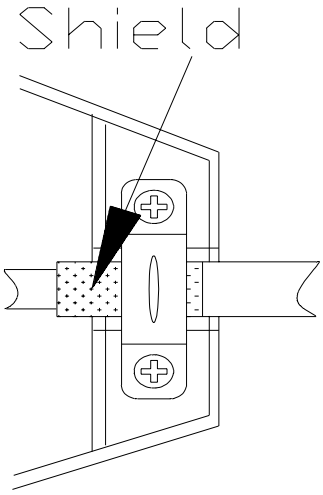
7.3.1 1st Free Programmable Interface via NET2 Connector

7.3.1.1 Terminals, Configuration

Note:

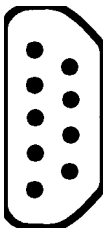
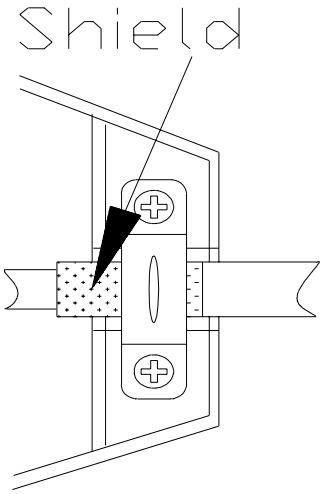
Either the JETWay via the NET2 connector or the Free Programmable Interface can be used. Simutaneously use of both interfaces is not possible.

Via NET2 connector

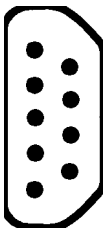
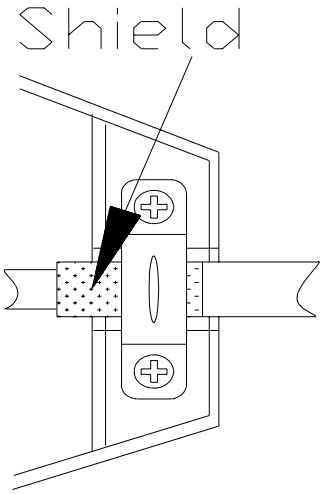
RS232 Cable Free Programmable Interface via NET2 Connector		
PROCESS-PLC		Specification max. Length
9 pin Sub-D-male 	 **Attach shield at both sides !** **Use metallised housings !**	RS232 max. cable length: 15m

Pin	Signal		Pin
2	TxD	RxD	
3	RxD	TxD	
7	Gnd		

Via NET2 connector

RS485 Cable (2 wire) Free Programmable Interface at the NET2 Connector		
PROCESS-PLC	**Shielding**	**Specification max. Length**
9 pin Sub-D-male 	 **Attach shield at both sides !** **Use metallised housings !**	RS485 max. cable length: 400m
Pin	**Signal**	**Remark**
7	Gnd	
8	Data +	
9	Data -	

Via NET2 connector

RS422/485 Cable (4 wire) Free Programmable Interface at the NET2 Connector			
PROZESS-PLC			Specification max. Length
9 pin Sub-D-male 	 **Attach shield at both sides !** **Use metallised housings !**		RS485 RS422 4 wire max. cable length: 400m
Pin	**Signal**		**Pin**
8	SDB	RDB	
9	SDA	RDA	
5	RDB	SDB	
6	RDA	SDA	

Set parameters

Register 62989: Configuration NET2: Free Programmable Interface	
Function	**Description**
Read	Current configuration Value after reset: 8
Write	New configuration: Bit0 0=free prog, 1=Net 2 Bit1 reserved Bit2 0=1, 1=2 stop bit Bit3 0=7, 1=8 bit / char Bit4,5 00=no parity 01=prohibited 10=odd 11=even Bit7,6 00=RS232 01=RS485 (4 wire) 10=RS422 11=RS485 (2 wire) **Default: PRIM, RS232, 8N1,9600**
Value range	0 .. 255

Remark:

Default setting: PRIM, RS232, 8N1, 9600

Remark:

PRIM = free programmable interface

7.3.1.2 Programming

Following programming example shows the handling of the free programmable interface.

Program listing

```
 0:       ; ************************************
 1:       ; * The program receives the ASCII     *
 2:       ; * characters 'A' to 'Z' converts     *
 3:       ; * them into small characters and     *
 4:       ; * sends them back                    *
 5:       ; *                                    *
 6:       ; ************************************
 7: DEF_FUNCTION [ReceiPRIM, RP]
      Par: rLimitTop, rLimitBottom
      Var: rHelp
 8:       ; ++++++++++++++++++++++++++++++++++++
 9:       ; +  Function reads a character from  +
10:       ; +  the reception buffer             +
11:       ; ++++++++++++++++++++++++++++++++++++
12:       REGISTER_LOAD [rHelp with R(rPRIMRecei)]  ;reads char from
                                                    ;buffer
13:       ; Check character for valid range
14:   IF
15:       LIMITS [Reg=rHelp, low=R(rLimitBottom),
                  up=R(rLimitTop)]
16:     THEN
17:       REGISTER_LOAD [rChar with R(rHelp)]       ;Char valid.
18:     ELSE
19:       REGZERO rChar                             ;Char not valid.
20:     THEN
21:       RETURN
22: END_DEF
23: DEF_FUNCTION [SendPRIM, SP]
      Par: rSendChar
24:       ; ++++++++++++++++++++++++++++++++++++
25:       ; +  The function write a character   +
26:       ; +  into the transmission buffer     +
27:       ; ++++++++++++++++++++++++++++++++++++
28:   IF
29:       REG rSendFill                             ;At least 1 char
30:       <                                         ;in buffer free ?
31:       128
32:     THEN
33:       REG rPRIMSend                             ;send modified
34:       =                                         ;received char
35:       REG rSendChar
36:       +
37:       32
38:     THEN
39:       RETURN
40: END_DEF
41: TASK tPRIMHandling --------------------------------
42:       REG rPRIMConfig                           ;Configuration
43:       =                                         ;RS422,
                                                     even Parity
44:       b00000000000000010111000                  ;8 bit, 1 stop bit.
45:       REGISTER_LOAD [rPRIMBaud with 7]          ;19200 baud.
46: LABEL sPRIMLoop
47:   IF
48:       NOT                                       ;1 or more chars
49:       REGZERO rReceiFill                        ;recepted ?
```

```
 50:     THEN
 51:       ReceiPRIM [rLimitTop=90, rLimitBottom=65]
 52:   IF
 53:       REGZERO rChar                              ;valid char
 54:     THEN                                         ;available ?
 55:       GOTO sPRIMLoop                             ;NO.
 56:     ELSE
 57:       SendPRIM [rSendChar=R(rChar)]
 58:       REGZERO rZeichen                           ;Confirmation.
 59:        sPRIMLoop
End of program
```

Symbol listing

```
**********  Task  **********
tPRIMhandling   0

**********  Labels  *********

MPRIMLoop       !


**********  Register  **************

rPRIMConfig     62989
rPRIMBaud       62990
rPRIMSend       62992    ;Transmission register
rPRIMEmpf       62993    ;Reception register
rReceiFill      61499    ;Reception buffer filling level
rSendFill       61502    ;Transmission buffer filling level
rChar           100
```

In the example above the actual transmission respectively reception is realized by functions.

Transmit character

A character is transmitted if the value is written into the register.

Receive character

A character is received if the value is read from the reception register. (Read access to register 62993 removes the character)

The reception buffer filling level is queried in register 61499.

The transmission buffer level is queried in register 61502.

7.3.1.3 Free Programmable Interface Registers (NET2)

Register 61499: Reception Buffer Filling Level	
Function	**Description**
Read	Current filling level of the reception buffer Value after reset: 0
Write	prohibited
Value range	0 .. 128

Register 61502: Transmission Buffer Filling Level	
Function	**Description**
Read	Current filling level of the transmission buffer Value after reset: 0
Write	prohibited
Value range	0 .. 128

Register 61508: Interface State	
Function	**Description**
Read	Current state Value after reset: state Interface state Bit0=1 one or more characters lost during reception Bit1=1 stop bit error Bit2=1 parity error Bit3=1 bit 0 to 2 is or was not 0 Bit4=1 reception buffer overflow Bit5=1 transmission buffer overflow
Write	prohibited
Value arnge	0 .. 128

Register 62989: Configuration NET2: Free Programmable Interface	
Function	**Description**
Read	Current configuration Value after reset: 8
Write	New configuration: Bit0=1 0=free prog, 1=Net 2 Bit1=1 reserved Bit2=1 0=1, 1=2 stop bit Bit3=1 0=7, 1=8 bit / char Bit4,5 00=no parity 01=prohibited 10=odd

	11 = even Bit7,6 00 = RS232 01 = RS485 (4 wire) 10 = RS422 11 = RS485 (2 wire) **Default: PRIM, RS232, 8N1,9600**
Value range	0 .. 255

Note:

If bit 0 = 1 (Reg 62989) all other bits are vain.

Register 62990: Baud Rate	
Function	**Description**
Read	Current baud rate Value after reset: 6
Write	New baud rate 0 150 1 300 2 600 3 1200 4 2400 5 4800 6 9600 default 7 19200 8 38400 9 57600 10 76800 11..14 115200
Value range	0 .. 255

Register 62992: Transmission Buffer	
Function	**Description**
Read	Last transmitted character Value after reset: 0
Write	Transmit character
Valuie range	0 .. 255

Note:

Transmission buffer size 128 characters.

Register 62993: Reception Buffer Access Removes Character	
Function	**Description**
Read	Received character Value after reset: 0
Write	Prohibited
Value range	0 .. 255

Note:

Reception buffer size 128 characters.

Register 62994: Reception Buffer Access does not Remove the Character	
Function	**Description**
Read	Received character Value after reset: 0
Write	Prohibited
Value range	0 .. 255

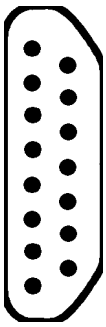
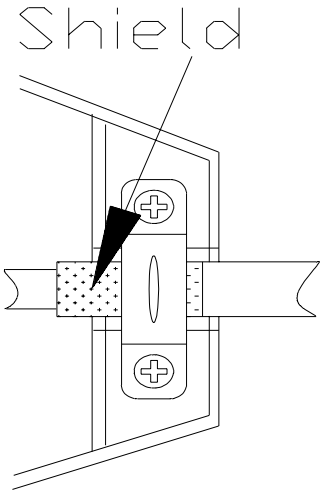
7.3.2 2nd Free Programmable Interface via LCD Connector

7.3.2.1 Terminals, Configuration

Note:

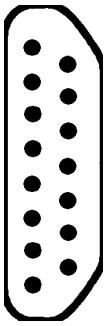
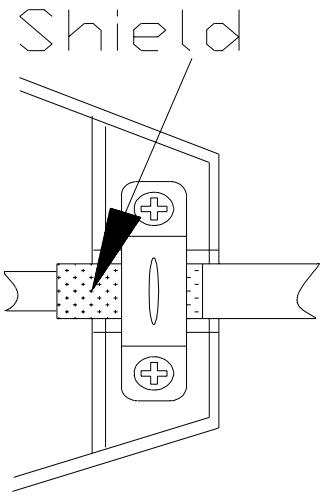
Either the User Interface via the LCD connector or the Free Programmable Interface can be used. Simutaneously use of both interfaces is not possible.

Via LCD connector

RS232 Cable Free Programmable Interface via LCD Connector		
PROCESS-PLC		**Specification max. Length**
15 pin Sub-D-male 	 **Attach shield at both sides !** **Use metallised housings !**	RS232 max. cable length: 15m

Pin	**Signal**		**Pin**
2	TxD	RxD	
3	RxD	TxD	
7	Gnd		

Via LCD connector

RS422 Cable (2 wire) Free Programmable Interface via LCD Connector			
PROZESS-PLC			
15 pin Sub-D-male 	 **Attach shield at both sides !** **Use metallised housings !**		RS485 RS422 4 wire max. cable length: 400m
Pin	**Signal**		**Pin**
7	Gnd	Gnd	
10	SDB	RDB	
11	SDA	RDA	
12	RDB	SDB	
13	RDA	SDA	

Set parameters

Register 63003: Configuration LCD: Free Programmable Interface	
Function	**Description**
Read	Current configuration Value after reset: 8
Write	New configuration: Bit0 0=free prog, 1=LCD Bit1 reserved Bit2 0=1, 1=2 stop bit Bit3 0=7, 1=8 bit / char Bit4,5 00=no parity 01=prohibited 10=odd 11=even Bit7,6 00=RS232 01=RS485 (4 wire) 10=RS422 11=RS485 (2 wire) **Default: PRIM, RS232, 8N1,9600**
Value range	0 .. 255

Remark:

Default setting: PRIM, RS232, 8N1, 9600

Remark:

PRIM = free programmable interface

7.3.2.2 Free Programmable Interface Registers (LCD)

Register 63002: Interface State	
Function	**Description**
Read	Current state Value after reset: state Interface state Bit3=1 character lost during reception, parity error, stop bit error Bit4=1 reception buffer overflow
Write	Resets error state
Value arnge	0 .. 24

Register 63003: Configuration LCD: Free Programmable Interface	
Function	**Description**
Read	Current configuration Value after reset: 8
Write	New configuration: Bit0=1 0=free prog, 1=Net 2 Bit1=1 reserved Bit2=1 0=1, 1=2 stop bit Bit3=1 0=7, 1=8 bit / char Bit4,5 00=no parity 01=prohibited 10=odd 11=even

	Bit7,6 00 = RS232 01 = RS485 (4 wire) 10 = RS422 11 = RS485 (2 wire) **Default: PRIM, RS232, 8N1,9600**
Value range	0 .. 255

Note:

If bit 0 = 1 (Reg 62989) all other bits are vain.

Register 62984: Baud Rate

Function	Description
Read	Current baud rate Value after reset: 6
Write	New baud rate 0 150 1 300 2 600 3 1200 4 2400 5 4800 6 9600 default 7 19200 8 38400
Value range	0 .. 255

Register 63004: Transmission Buffer

Function	Description
Read	Last transmitted character Value after reset: 0
Write	Transmit character
Valuie range	0 .. 255

Note:

Transmission buffer size 255 characters.

Register 63005: Reception Buffer Access Removes Character

Function	Description
Read	Received character Value after reset: 0
Write	Prohibited
Value range	0 .. 255

Note:

Reception buffer size 255 characters.

Register 63006: Reception Buffer Access does not Remove the Character

Function	Description
Read	Received character Value after reset: 0
Write	Prohibited
Value range	0 .. 255

Register 62155: Reception Buffer Filling Level

Function	Description
Read	Current filling level of the reception buffer Value after reset: 0
Write	prohibited
Value range	0 .. 128

Register 62156: Transmission Buffer Filling Level

Function	Description
Read	Current filling level of the transmission buffer Value after reset: 0
Write	prohibited
Value range	0 .. 128

7.4 Realtime Clock

7.4.1 Overview, Function

A realtime clock is integrated into the PROCESS-PLC DELTA. This real time clock is nonvolatile and not dependent of the RAM buffer battery.

There are 2 register sets with 8 registers each. Register set 1 (62920 bis 62927) allows read and write access. Write access to this registers set the time directly. Read accesses read the time.

Besides, there is register set 2 (62912 bis 62919). This set has the following meaning: If a program waits for a certain time, no change of the operands (time...) of the comparison must be guaranteed. Therefore read access to set 1 copies all data to set 2. There they remain unchanged until a new read access to register 1 occurs (see example).

To set the clock the values are written into set 2 and completely transfered to the real time clock by 1 write access to set 1.

7.4.2 Register Description

Register Set1 read/write direct	Register Set2 read/write buffer	Data	Range
62920	62912	Seconds	0-59
62921	62913	Minutes	0-59
62922	62914	Hours	0-23
62923	62915	12/24h Format	0,128
62924	62916	Day of week	1-7
62925	62917	Day (date)	1-31
62926	62918	Month	1-12
62927	62919	Year	0-99

Reg 62924 Day of week

Register **62924** has got the following function: the register content is **Day of Week**. 1=Sunday, 2=Monday, 3=Thuesday etc.

Reg 62923 selects am <-> pm

Register **62923** selects between **am** and **pm**. Before 12 hour mode has to be selected by bit 1 of register 63240.

To display the time in the used format **2** has to be written into special register **61454**. If this register contains the value 2 the sign place of a DISPLAY_REG instruction is suppressed (see example).

Register 63240: Realtime Clock Configuration	
Function	**Description**
Read	Current RTC configuration Value after reset: 0
Write	Bitcoded register: Bit 0=1 summer/winter time Bit 1=0 am/pm format

	Bit 1 = 1 24 hour format
Value range	0 .. 255

7.4.3 Realtime Clock Example Program

The following example program displays the real time clock data on the user interface.

```
0: TASK 0 ----------------------------------------
 1:       ;
 2:       REGISTER_LOAD [61454 with 2]            ;no sign
   3:       FLAG 2055
;leading zeros
 4:       REGISTER_LOAD [61453 with 6]     ;2 places number
 5:       DISPLAY_TEXT [#0, cp=1, "_Current time:"]
 6:       ;
 7: LABEL 100
 8:       CALL 900
 9:       DELAY 5
10:       GOTO 100
11:       ;
12: LABEL 900                                      ;->
DISPLAY
13:       DISPLAY_REG [#0, cp=25, reg=62925]    ;store time
14:       DISPLAY_TEXT [#0, cp=27, ".  .19  ,  :  :  "]
15:       DISPLAY_REG [#0, cp=28, reg=62918]      ;month
16:       DISPLAY_REG [#0, cp=33, reg=62919]      ;year
17:       ;
18:       ; ------- Display time -------
19:       ;
20:       DISPLAY_REG [#0, cp=36, reg=62914]      ;hour
21:       DISPLAY_REG [#0, cp=39, reg=62913]      ;minute
22:       DISPLAY_REG [#0, cp=42, reg=62912]      ;seconds
23:       RETURN
End of program
```

7.5 Operating System Update (Download)

Operating system updates can be realized via the **Transfer** menu of the SYMPAS programming environment.
For this JETTER makes available operating system file on their mailbox (**extension: *.OS**).

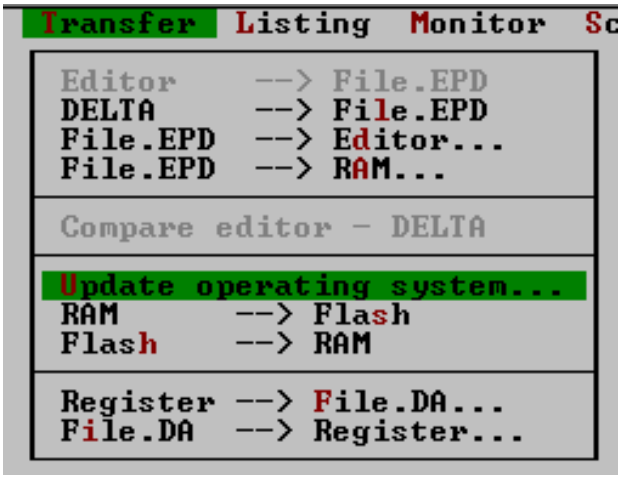


Figure 17: Operating system updates can be realized by the SYMPAS programming environment

Note:

The timeout time has to be set to 4000 in the **Special/settings** menu for operating system updates.

Index