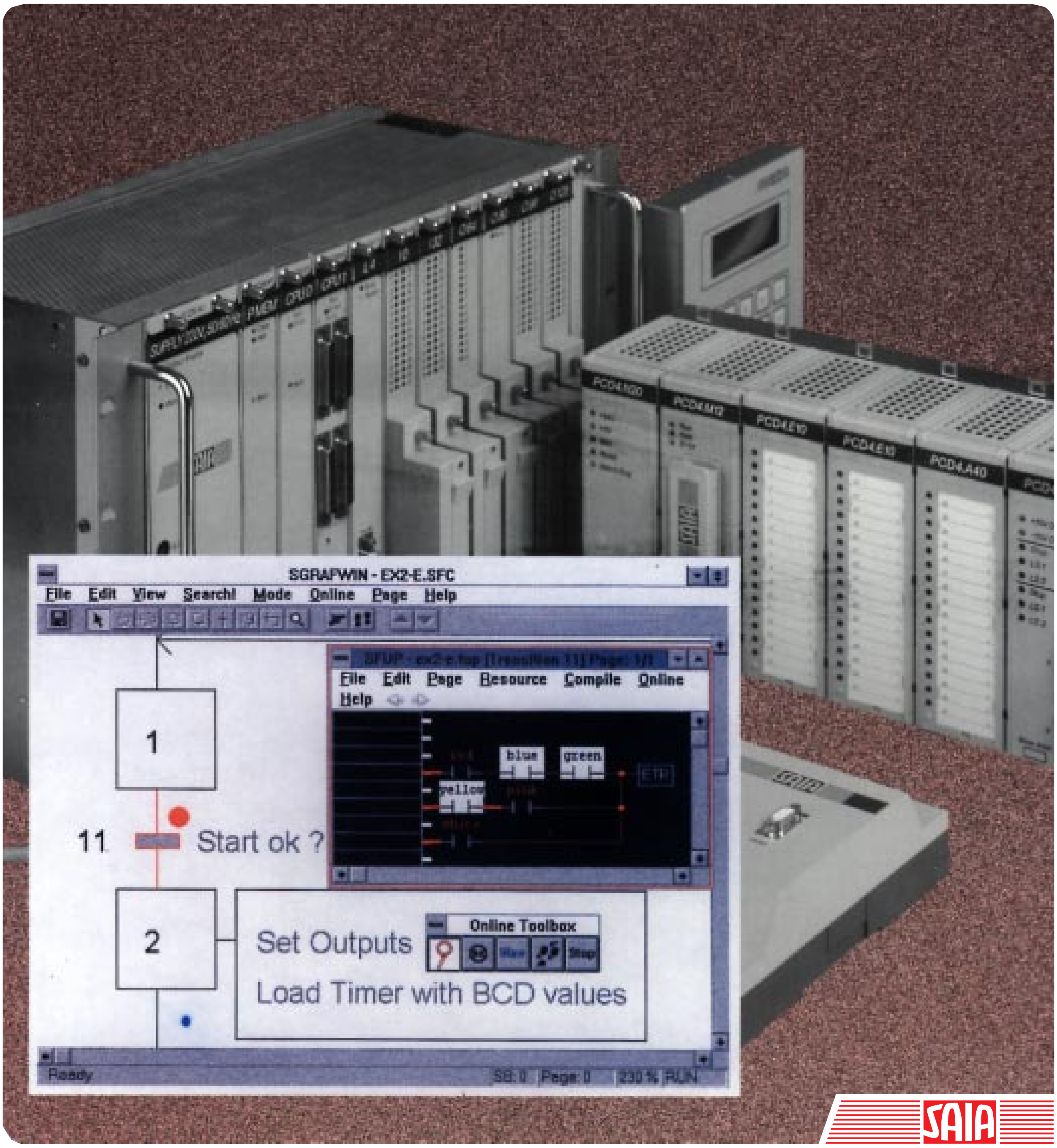
SAIA-Burgess Electronics

SWITCHES • MOTORS • CONTROLLERS

SAIA®PCD
Process Control Devices

The FUPLA and the KOPLA function families


SGRAPWIN - EX2-E.SFC
File Edit View Search! Mode Online Page Help
File Edit Page Resource Compile Online Help
blue
green
yellow
Online Toolbox
1
11
Start ok ?
2
Set Outputs
Load Timer with BCD values
Ready
SB:0 Page:0 230 % RUN

Edition 26/749 E1

SAIA-Burgess Electronics Ltd.
Bahnhofstrasse 18
CH-3280 Murten (Switzerland)
http;//www.saia-burgess.com

BA: Electronic Controllers   Telephone 026 / 672 72 72
                              Telefax 026 / 672 74 99

SAIA-Burgess Companies

Switzerland
SAIA-Burgess Electronics AG
Freiburgstrasse 33
CH-3280 Murten
☎ 026 672 77 77, Fax 026 670 19 83

France
SAIA-Burgess Electronics Sàrl.
10, Bld. Louise Michel
F-92230 Gennevilliers
☎ 01 46 88 07 70, Fax 01 46 88 07 99

Germany
SAIA-Burgess Electronics GmbH
Daimlerstrasse 1k
D-63303 Dreieich
☎ 06103 89 060, Fax 06103 89 06 66

Nederlands
SAIA-Burgess Electronics B.V.
Hanzeweg 12c
NL-2803 MC Gouda
☎ 0182 54 31 54, Fax 0182 54 31 51

Austria
SAIA-Burgess Electronics Ges.m.b.H.
Schallmooser Hauptstrasse 38
A-5020 Salzburg
☎ 0662 88 49 10, Fax 0662 88 49 10 11

Belgium
SAIA-Burgess Electronics Belgium
Avenue Roi Albert 1er, 50
B-1780 Wemmel
☎ 02 456 06 20, Fax 02 460 50 44

Italy
SAIA-Burgess Electronics S.r.l.
Via Cadamosto 3
I-20094 Corsico MI
☎ 02 48 69 21, Fax 02 48 60 06 92

Hungary
SAIA-Burgess Electronics Automation Kft.
Liget utca 1.
H-2040 Budaörs
☎ 23 501 170, Fax 23 501 180

Representatives

Great Britain
Canham Controls Ltd.
25 Fenlake Business Centre, Fengate
Peterborough PE1 5BQ UK
☎ 01733 89 44 89, Fax 01733 89 44 88

Portugal
INFOCONTROL Electronica e Automatismo LDA.
Praceta Cesário Verde, No 10 s/cv, Massamá
P-2745 Queluz
☎ 21 430 08 24, Fax 21 430 08 04

Denmark
Malthe Winje Automation AS
Håndværkerbyen 57 B
DK-2670 Greve
☎ 70 20 52 01, Fax 70 20 52 02

Spain
Tecnosistemas Medioambientales, S.L.
Poligono Industrial El Cabril, 9
E-28864 Ajalvir, Madrid
☎ 91 884 47 93, Fax 91 884 40 72

Norway
Malthe Winje Automasjon AS
Haukelivn 48
N-1415 Oppegård
☎ 66 99 61 00, Fax 66 99 61 01

Czech Republic
ICS Industrie Control Service, s.r.o.
Modranská 43
CZ-14700 Praha 4
☎ 2 44 06 22 79, Fax 2 44 46 08 57

Sweden
Malthe Winje Automation AB
Truckvägen 14A
S-194 52 Upplands Våsby
☎ 08 795 59 10, Fax 08 795 59 20

Poland
SABUR Ltd.
ul. Druzynowa 3A
PL-02-590 Warszawa
☎ 22 844 63 70, Fax 22 844 75 20

Suomi/ Finland
ENERGEL OY
Atomitie 1
FIN-00370 Helsinki
☎ 09 586 2066, Fax 09 586 2046

Australia
Siemens Building Technologies Pty. Ltd.
Landis & Staefa Division
411 Ferntree Gully Road
AUS-Mount Waverley, 3149 Victoria
☎ 3 9544 2322, Fax 3 9543 8106

Argentina
MURTEN S.r.l.
Av. del Libertador 184, 4° "A"
RA-1001 Buenos Aires
☎ 054 11 4312 0172, Fax 054 11 4312 0172

After sales service

USA
SAIA-Burgess Electronics Inc.
1335 Barclay Boulevard
Buffalo Grove, IL 60089, USA
☎ 847 215 96 00, Fax 847 215 96 06

Issue : 22.11.2000

SAIA® Process Control Devices

Programming tools for MS WINDOWS

The FUPLA and the KOPLA function families

PG4 - Version 1.3

SAIA-Burgess Electronics Ltd. 1996.   All rights reserved
Edition 26/749 E1 - 04.1996

Subject to technical changes

Updates

Manual :
The FUPLA and the KOPLA function families - PG4 Version 1.3 - Edition E1

Date	Chapter	Page	Description
27.10.2000	---	---	Small updates for the “Support Homepage”

Contents

In this manual all functions of the standard FUPLA and the KOPLA are described.

This manual represents the chapters 4.4 and 4.5 of the manual "Programming Tools PG4" (26/748 E)

The descriptions of the functions are normally identical to the "Infos" of the functions on the screen.

Overview over the function families **Page**

Notes :

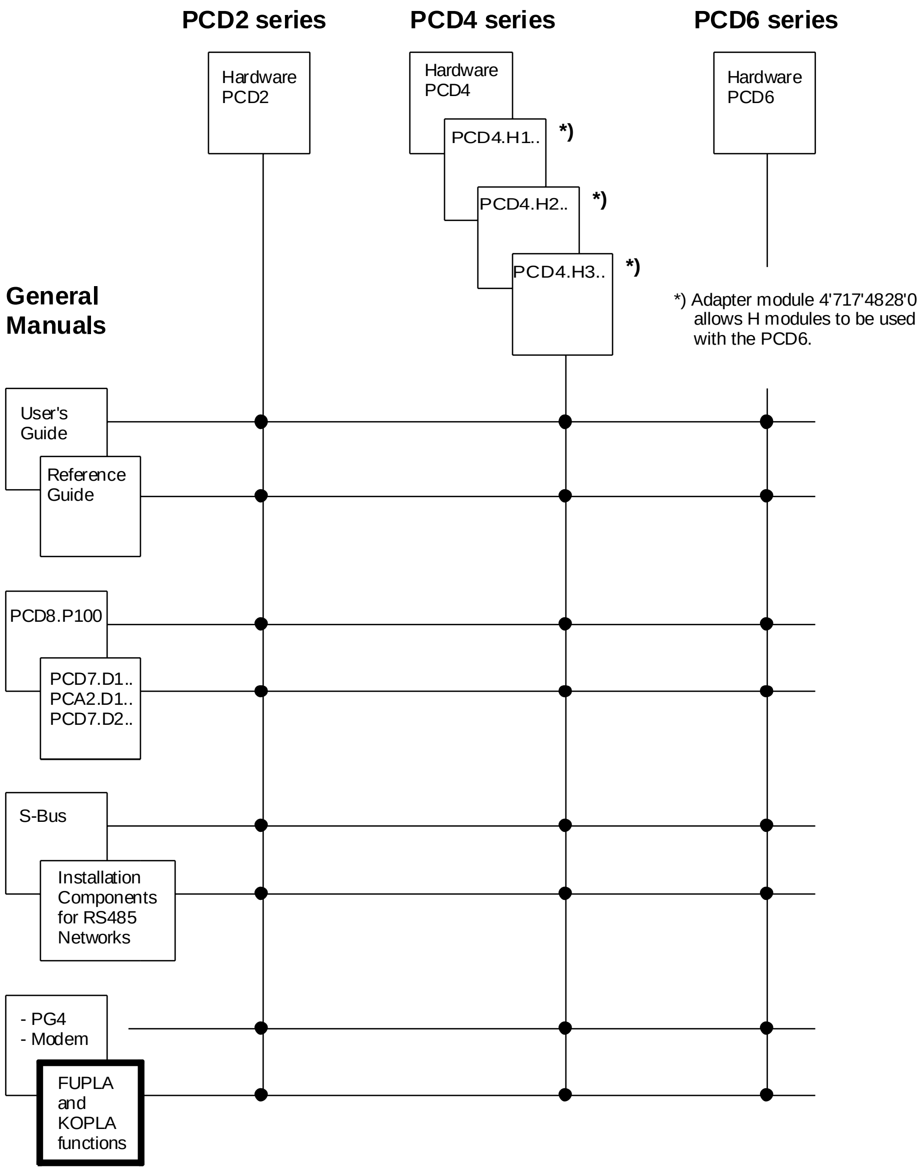
Please note:

A number of detailed manuals are available to aid installation and operation of the SAIA PCD. These are for use by technically qualified staff, who may also have successfully completed one of our "workshops".

To obtain the best performance from your SAIA PCD, closely follow the guidelines for assembly, wiring, programming and commissioning given in these manuals. In this way, you will also become one of the many enthusiastic SAIA PCD users.

If you have any technical suggestions or recommendations for improvements to the manuals, please let us know. A form is provided on the last page of this manual for your comments.

Summary



Reliability and safety of electronic controllers

SAIA AG is a company which devotes the greatest care to the design, development and manufacture of its products:

- state-of-the-art technology
- compliance with standards
- ISO 9001 certification
- international approvals: e.g. Germanischer Lloyd,
  Det Norske Veritas, CE mark ...
- choice of high-quality componentry
- quality control checks at various stages of production
- in-circuit tests
- run-in (burn-in at 85°C for 48h)

Despite every care, the excellent quality which results from this does have its limits. It is therefore necessary, for example, to reckon with the natural failure of components. For this reason SAIA AG provides a guarantee according to the "General terms and conditions of supply".

The plant engineer must in turn also contribute his share to the reliable operation of an installation. He is therefore responsible for ensuring that controller use conforms to the technical data and that no excessive stresses are placed on it, e.g. with regard to temperature ranges, overvoltages and noise fields or mechanical stresses.

In addition, the plant engineer is also responsible for ensuring that a faulty product in no case leads to personal injury or even death, nor to the damage or destruction of property. The relevant safety regulations should always be observed. Dangerous faults must be recognized by additional measures and any consequences prevented. For example, outputs which are important for safety should lead back to inputs and be monitored from software. Consistent use should be made of the diagnostic elements of the PCD, such as the watchdog, exception organization blocks (XOB) and test or diagnostic instructions.

If all these points are taken into consideration, the SAIA PCD will provide you with a modern, safe programmable controller to control, regulate and monitor your installation with reliability for many years.

4.4 The function families of the FUPLA

Overview of functions in the individual function families (arranged according to function and usage *))

*) In FUPLA all functions are automatically arranged in alphabetical order.

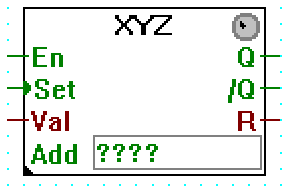
4.4.6 Integer arithmetic

4.4.7 Floating point arithmetic

Explanation of format and symbols used in function descriptions:

Function name

[_fupxyz]



Function name

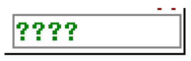
Name of function from the FUPLA menu (from FBox Selection)

On the left of the FBox are the inputs, e.g.:

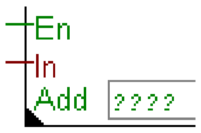
Inputs:	– En	→	Enable	"–"	binary input (H/L)
	> Set	→	Set	">"	dynamic binary input
	= Val	→	Value	"="	numeric input (value)

On the right of the FBox are the outputs, e.g.:

Outputs:	– Q	→	Output	"–"	binary output (H/L)
	– /Q	→	Output inv.	"–"	binary output (H/L)
	= R	→	Result	"="	numeric output (value)

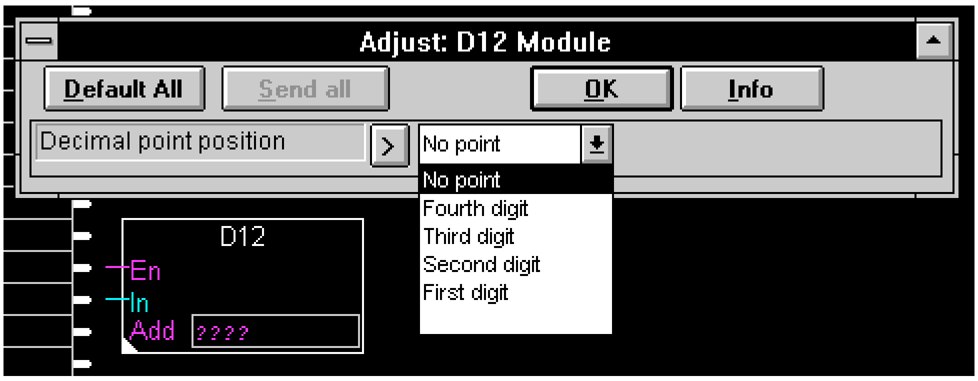


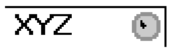
Field for the base address of the function, normally "O" or "o" (output) is to be entered, e.g. "o 16"



The dot in the left lower corner indicates that this FBox has an adjust window. This adjust window can be opened by double clicking on the FBox. The arrow symbol must be selected.

Example for a adjust window: 'Display' - 'D12 Module'.



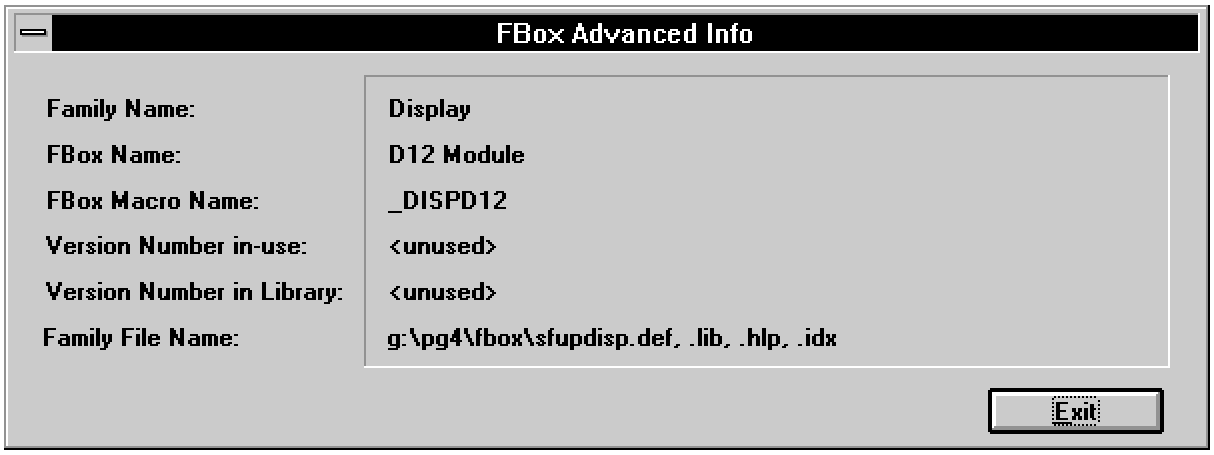


More complex functions, e.g. communications, contain an "LED" in the FBox. This remains grey during programming. In RUN, if everything is okay, the LED becomes green. If there is an error or is switched to manual, the LED is red.

[fupabc] (on the title bar of each function family) shows the file names of this family, e.g. [sfuptime] for the time related function family: "sfuptime.def", "sfuptime.hlp", "sfuptime.idx", "sfuptime.lib".

[_fupxyz]: Function name within the function family, e.g. [_ondel] for the delayed switching on in the "sfuptime.xxx" family.

The internal name of the function is shown after clicking on 'Advanced Info':


FBox Advanced Info

Family Name:	Display
FBox Name:	D12 Module
FBox Macro Name:	_DISPD12
Version Number in-use:	<unused>
Version Number in Library:	<unused>
Family File Name:	g:\pg4\fbox\sfupdisp.def, .lib, .hlp, .idx

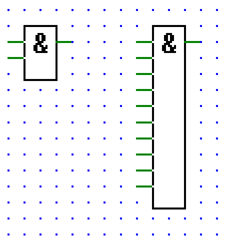
Notes :

4.4.1 Binary functions [fupbina]

4.4.1.1 And 2-10 inputs

And 2-10 inputs

[_band]



Inputs/Outputs: – all → binary format.

Outputs the binary AND of its inputs. The output is high only if all inputs are high.

Stretchable from 2 to 10 inputs.

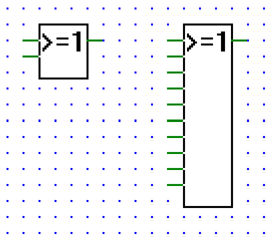
E.g.: 2 input And:

```
 In1 | In2 | Out
-----+-----+----
  0  |  0  |  0
  0  |  1  |  0
  1  |  0  |  0
  1  |  1  |  1
```

4.4.1.2 Or 2-10 inputs

Or 2-10 inputs

[_bor]



Inputs/Outputs: – all → binary format.

Outputs the binary OR of its inputs.

The output is high when at least one of its inputs is high.

Stretchable from 2 to 10 inputs.

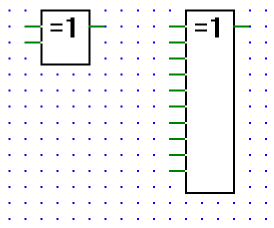
E.g.: 2 inputs OR:

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	1

4.4.1.3 Xor 2-10 inputs

Xor 2-10 inputs

[_bxor]



Inputs/Outputs: – all → binary format.

Outputs the binary XOR of its inputs.

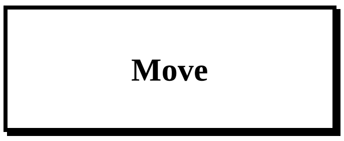
The output is high when only one of its inputs is high.

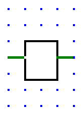
Stretchable from 2 to 10 inputs.

E.g.: XOR with 2 inputs:

In1	In2	Output
0	0	0
0	1	1
1	0	1
1	1	0

4.4.1.4 Move



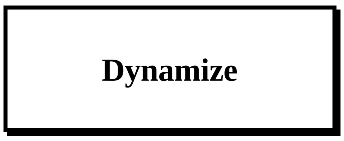


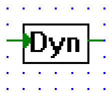
[_bmove]

Inputs/Outputs: – all → binary format.

Outputs its binary input, connects input labels directly to output labels.

4.4.1.5 Dynamize





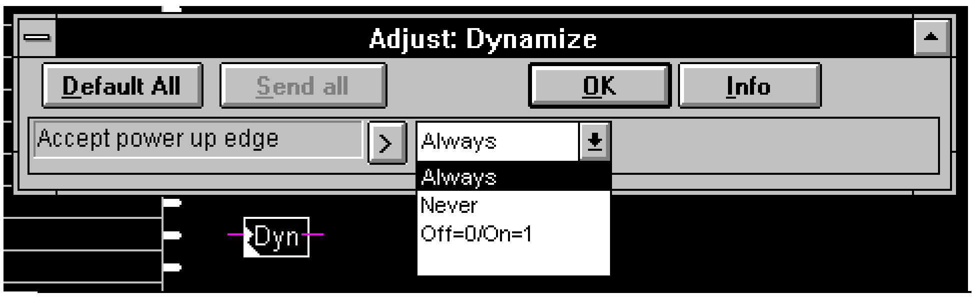
[_bdyn2]

Inputs/Outputs: – all → binary format.

Detects rising edges.

The output is High only when the input has gone from Low to High.

3 options are available to accept edges at power up if the input is already High.



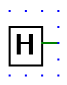
Accept power up edge: Always

"Always": Input High is always considered as a rising edge at power up.
"Never": Input High is never considered as a rising edge at power up.
"Off=0/On=1": A rising edge is only considered if the input was Low at power down and is High at power up.

Warning: The last option can produce undefined reactions if the Flags are not reset after a modification of the program!

4.4.1.6 High

High



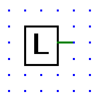
[_bhigh]

Output: – → binary format.

Outputs a binary high state.

4.4.1.7 Low

Low



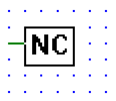
[_blow]

Output: – all → binary format.

Outputs a binary low.

4.4.1.8 Not connected

Not connected



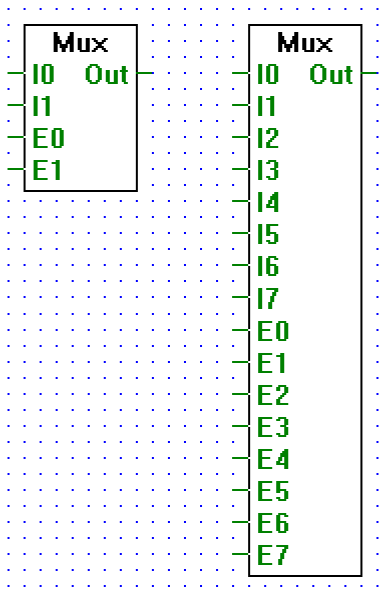
[_bnotcn]

The "Not Connected" box, terminates unused binary outputs.

4.4.1.9 Multiplexer with binary selection

Mux binary selection

[_bmux]



Inputs/Outputs: – all → binary format.

Transfers an I0..I7 to an output when the corresponding enable signal E0..E7 is high.

Stretchable from 2 to 8 inputs.

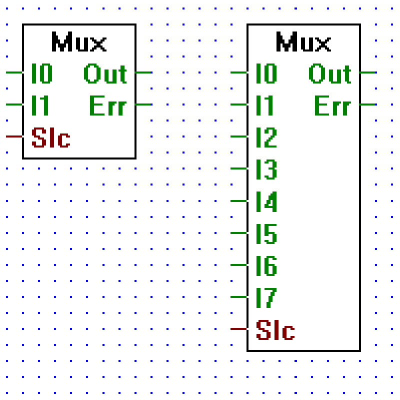
If all enable inputs are low, a low is output.

The enable line E0 has the highest priority and E7 has the lowest. Therefore if one or more enable lines are actived at the same time, the input associated with the highest priority enable line will be output.

4.4.1.10 Multiplexer with integer selection

Mux integer selection

[_bmux2]



Inputs:	– I0:	
	¦	binary format
	– I7:	
	= Slc:	Selection: integer format
Outputs:	– Out:	binary format
	– Err:	binary format

Outputs to "Out" the state of the input "In" selected via the input "Slc" (e.g. when Slc == 0 outputs the state of I0, when Slc == 5 outputs the state of I5...).

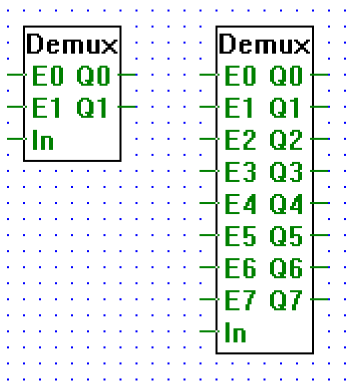
Stretchable from 2 to 8 inputs "In".

When "Slc" is out of range (i.e. "Slc" < 0 or "Slc" > n), "Out" is set low and "Err" is set high.

4.4.1.11 Demultiplexer with binary selection

Demux binary selection

[_bdemux]



Inputs/Outputs: – all → binary format.

Transfers binary input In to an output (Q0..Q7) when its corresponding enable line (E0..E7) is high.

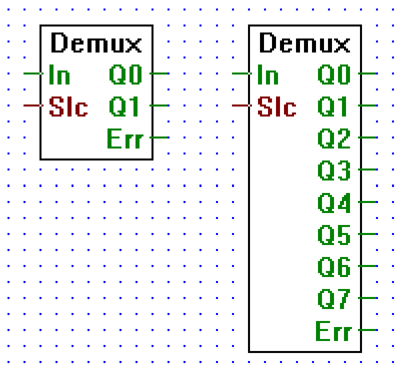
Stretchable from 2 to 8 inputs.

When an enable line (E0..E7) is low, its corresponding output (Q0..Q7) is set low.

4.4.1.12 Demultiplexer with integer selection

Demux integer selection

[_bdemux2]



Inputs:	– In:	binary format
	= Slc:	Selection: integer format
Outputs:	– Q0:	binary format
	¦	
	– Q7:	binary format
	– Err:	binary format

Transfers binary input "In" to a selected output (Q0..Qn). Outputs are selected via the input "Slc". E.g. when "Slc" == 1 the input "In" is transfered to "Q1".

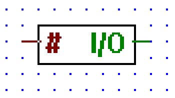
When not selected an output is set low.

Stretchable from 2 to 8 outputs.

Err goes high only when "Slc" is out of range, i.e. when "Slc" < 0 or when "Slc" > n.

4.4.1.13 I/O indirect

I/O.indirect



[_bioind]

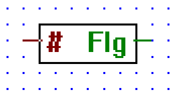
Input: = I/O adresse: integer format
Output: – I/O state: binary format

Outputs the state of I/O number #.

E.g.: if the # input is 5, then the state of I/O 5 is output.

4.4.1.14 Flag indirect

Flag indirect



[_bflgin]

Input: = Flag adresse: integer format
Output: – Flag state: binary format

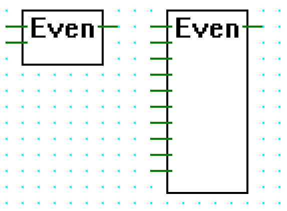
Outputs the state of flag number #.

E.g.: if the # input is 5, then the state of the flag 5 is output.

4.4.1.15 Even, 2-10 inputs

Even 2-10 inputs

[_beven]



Inputs: – binary format
Output: – binary format

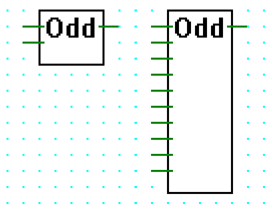
The Output is High if the number of Inputs High is Even. Otherwise, the Output is Low.

E.g. 3 inputs:

In1	In2	In3	Out
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

4.4.1.16 Odd, 2-10 inputs

Odd 2-10 inputs



[_bodd]

Inputs:	–	binary format
Output:	–	binary format

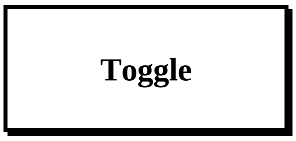
The Output is High if the number of Inputs High is Odd. Otherwise, the Output is Low.

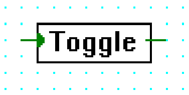
E.g. 3 inputs:

In1	In2	In3	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

4.4.2 Flip-Flops [sfupflip]

4.4.2.1 Toggle





[_flitog]

Inputs/Outputs: – all → binary format.

Simple toggle function (Flip-Flop).

The state of the output is toggled at each positive edge of the input.

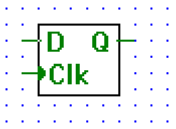
On start up, the output is initialized to Low. If the input is High on start up, the output is set to High.

Einfache Umschaltfunktion (toggle).

```
Input    __---__-__----__----
Output   __-----___------____
```

4.4.2.2 Type D

Type D



[_flid]

Inputs/Outputs: – all → binary format.

D type flip-flop with a rising edge triggered clk input.

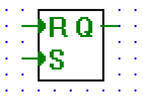
The state of the input D is stored when the clk input goes from the low to high.

The output Q is always the last stored state of input D.

```
D   _____------_-__--___
Clk __--__--__--__--__--
Q   ______--------______
```

4.4.2.3 Type RS dynamized

Type RS dynamized



[_flirs]

Inputs/Outputs: – all → binary format.

RS flip-flop, with priority on the R input. Both R and S are rising edge triggered.

When input R goes from low to high, output Q is set low.

When input S goes from low to high, output Q is set high.

When both inputs R and S go from low to high at the same time, output Q is set low.

In all the other cases the output is unchanged.

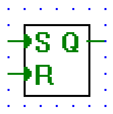
```
    R ___--_________---_____--___
    S ________---______---__---__
    Q ________------__-----______

  R | S | Q
 ---+---+---
  X | X | Unchanged
  X |_/-| 1
 _/-| X | 0
 _/-|_/-| 0
```

_/-	=	rising edge.
X	=	any state.

4.4.2.4 Type SR dynamized

Type SR dynamized



[_flisr]

Inputs/Outputs: – all → binary format.

RS flip-flop, with priority on the S input. Both R and S are rising edge triggered.

When input R goes from low to high, output Q is set low.

When input S goes from low to high, output Q is set high.

When both inputs R and S go from low to high at the same time, output Q is set high.

In all the other cases the output is unchanged.

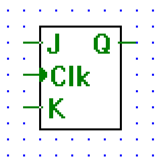
```
 R _______---_____---__--___
 S ___--________---_____---__
 Q ___----______--_____-----

 R | S | Q
---+---+---
 X | X | Unchanged
 X |_/-| 1
_/-| X | 0
_/-|_/-| 1
```

_/-	=	rising edge.
X	=	any state.

4.4.2.5 Type JK

Type JK



[_flijk]

Inputs/Outputs: – all → binary format.

Clocked JK flip-flop.

The J and K inputs are read only when a rising edge is detected on the Clk input. When no rising edge is detected, the output Q is unchanged.

When a rising edge is detected on the Clk input:

- and J is high, Q is reset low,
- and K is high, Q is set high,
- and J and K are high, Q is toggled,
- and J and K are low, Q is unchanged.

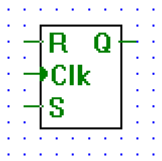
```
Clk __--__--__--__--__--_
J   _______-_-------_---_
K   ____----_____--_----_
Q   ______----____----___
```

Clk	J	K	Q(t)
X	X	X	Q(t-1)
_/-	0	0	Q(t-1)
_/-	0	1	1
_/-	1	0	0
_/-	1	1	/Q(t-1) (toggle)

_/-	=	rising edge.
X	=	any state.

4.4.2.6 Type RS clocked

Type RS clocked



[_flirsclk]

Inputs/Outputs: – all → binary format.

Clocked RS flip-flop, with priority on the R input.

Inputs R and S are read when a rising edge is detected on input Clk.
When no rising edge is detected, output Q is unchanged.

When a rising edge is detected at input Clk:

- and R is high, Q is reset low,
- and S is high, Q is set high,
- and R and S are high, Q is reset low,
- and R and S are low, Q is unchanged.

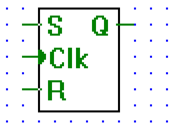
```
Clk __--__--__--__--__--_
  R   _______-_---_____---_
  S   ____--------_---_____
  Q   ______----____----___


  Clk | R | S | Q
  ----+---+---+---
   X  | X | X | Unchanged
  _/- | 0 | 0 | Unchanged
  _/- | 0 | 1 | 1
  _/- | 1 | 0 | 0
  _/- | 1 | 1 | 0
```

_/-	=	rising edge.
X	=	any state.

4.4.2.7 Type SR clocked

Type SR clocked



[_flisrclk]

Inputs/Outputs: – all → binary format.

Clocked RS flip-flop, with priority on the S input.

Inputs R and S are read when a rising edge is detected on input Clk.
When no rising edge is detected, output Q is unchanged.

When a rising edge is detected at input Clk:

- and R is high, Q is reset low,
- and S is high, Q is set high,
- and R and S are high, Q is set high,
- and R and S are low, Q is unchanged.

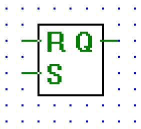
```
Clk __--__--__--__--_
R   _____--------_---_
S   _--_____-_---______
Q   __----_____----____
```

Clk	R	S	Q
X	X	X	Unchanged
_/-	0	0	Unchanged
_/-	0	1	1
_/-	1	0	0
_/-	1	1	1

_/-	=	rising edge.
X	=	any state.

4.4.2.8 Type RS

Type RS



[_flirsndyn]

Inputs/Outputs: – all → binary format.

RS flip-flop, with priority on the R input.

When input R is high, output Q is set low.

When input S is high and input R is low, output Q is set high.

When both inputs R and S are high, output Q is set low.

In all the other cases the output is unchanged.

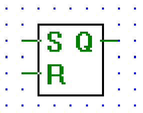
```
R ___--________---____--___
S _______---_____---___---_
Q _______------___----__---
```

R	S	Q
0	0	Unchanged
1	X	0
0	1	1

X = any state.

4.4.2.9 Type SR

Type SR



[_flisrndyn]

Inputs/Outputs: – all → binary format.

RS flip-flop, with priority on the S input.

When input R is high and input S is low, output Q is set low.

When input S is high, output Q is set high.

When both inputs R and S are high, output Q is set high.

In all the other cases the output is unchanged.

```
R _______---_____---___---__
S ___--________---____---___
Q ___----______---____---___
```

R	S	Q
0	0	Unchanged
X	1	1
1	0	0

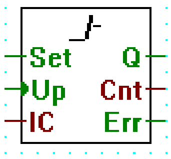
X = any state.

Notes :

4.4.3 Counters [sfupcoun]

4.4.3.1 Up count with preset

Up with preset



[_uppr2]

Inputs:	– Set:		binary format
	> Up:		binary format
	= IC	Initial Count:	integer format
Outputs:	– Q:	pos/neg	binary format
	= Cnt:	Count	integer format
	– Err:	Error	binary format

Increments a counter each time a rising edge is detected on input Up. The count is output at Cnt.

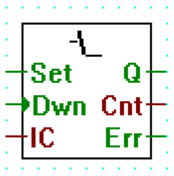
The Set input loads the counter with value IC.

If the counter is > 0, the output Q is High and if it is <= 0, the output Q is Low.

If the counter overflows, output Err is set high.

4.4.3.2 Down count with preset

Down with preset



[_dwnpr2]

Inputs:	– Set:		binary format
	> Dwn:	Down:	binary format
	= IC:	Initial Count:	integer format

Outputs:	– Q:	pos/neg	binary format
	= Cnt:	Count	integer format
	– Err:	Error	binary format

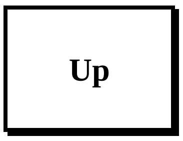
Decrements a counter each time a rising edge is detected on input Dwn. The count is output at Cnt.

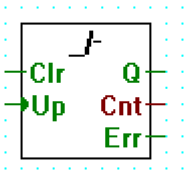
The Set input loads the counter with value IC.

If the counter is > 0, the output Q is High and if it is <= 0, the output Q is Low.

If the counter underflows, output Err is set high.

4.4.3.3 Up count





[_up2]

Inputs:	– Clr:	Clear	binary format
	> Up:		binary format
Outputs:	– Q:	pos/neg	binary format
	= Cnt:	Count	integer format
	– Err:	Error	binary format

Increments a counter each time a rising edge is detected on input Up. The count is output at Cnt.

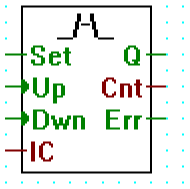
The Clr input clears the counter.

If the counter is > 0, the output Q is High and if it is <= 0, the output Q is Low.

If the counter overflows, output Err is set high.

4.4.3.4 Up/down count with preset

Up/down with preset



[_updnpr2]

Inputs:	– Set:		binary format
	> Up:		binary format
	> Dwn:		binary format
	= IC:	Initial Count:	integer format
Outputs:	– Q:	pos/neg	binary format
	= Cnt:	Count	integer format
	– Err:	Error	binary format

Increments or decrements a counter each time a rising edge is detected on the corresponding input Up or Dwn. The count is output at Cnt.

The Set input loads the counter with value IC.

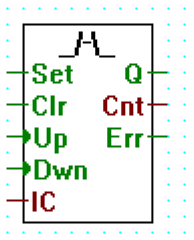
If the counter is > 0, the output Q is High and if it is <= 0, the output Q is Low.

If the counter overflows or underflows, output Err is set high.

4.4.3.5 Up/down count with preset and clear

Up/down preset and clear

[_updnsc]



Inputs:	– Set:		binary format
	– Clr		binary format
	> Up:		binary format
	> Dwn:		binary format
	= IC:	Initial Count:	integer format
Outputs:	– Q:	pos/neg	binary format
	= Cnt:	Count	integer format
	– Err:	Error	binary format

Increments or decrements a counter each time a rising edge is detected on the corresponding input Up or Dwn. The count is output at Cnt.

The Set input loads the counter with value IC.

The Clr input clears the counter.

If the counter is > 0, the output Q is High and if it is <= 0, the output Q is Low.

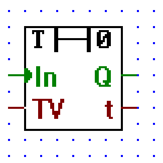
If the counter overflows or underflows, output Err is set high.

Notes :

4.4.4 Time related functions [sfuptime]

4.4.4.1 On delay

On delay



[_ondel]

Inputs:	> In:	Activate	binary format
	= TV:	Timer Value:	integer format
Outputs:	– Q	Output	binary format
	= t:	actual value	integer format

On-delay timer.

Q is set high at TV timer units after In was set high. When In goes low, Q is immediately set low.

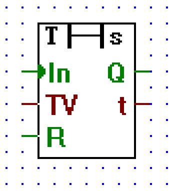
If In goes high for a period smaller than TV, then Q is unaffected. t indicates the value of the internal timer.

E.g.

```
In   ___-------___--__------__
Q    _______---___________--__
        <TV>          <TV>
```

4.4.4.2 Store delay

Store delay



[_stodel]

Inputs:	> In:	Activate	binary format
	= TV:	Timer Value:	integer format
	– R:	Reset	binary format
Outputs:	– Q	Output	binary format
	= t:	actual value	integer format

Store on-delay timer.

When In goes high, Q is set high at TV timer units later.

Once Q is set high it goes low only when R (reset) is activated (set high). When R is activated, Q is held low no matter the state of In.

If In goes to high for a period smaller than TV then Q is unaffected.

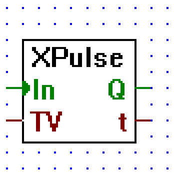
t indicates the value of the internal timer.

E.g.

```
In ___------____--__------__--------
R  __________--_______________------
Q  _______---___________--------____
      <TV>          <TV>
```

4.4.4.3 Exclusive pulse

Exclusive pulse



[_xpulse]

Inputs:	> In:	Activate	binary format
	= TV:	Timer Value:	integer format
Outputs:	– Q	Output	binary format
	= t:	actual value	integer format

Extended pulse timer, generates a pulse of duration TV when it detects a rising edge on the input.

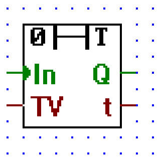
t indicates the value of the internal timer.

E.g.

```
In__------__--____
Q __----____----__
    <TV>    <TV>
```

4.4.4.4 Off delay

Off delay



[_offdel]

Inputs:	> In:	Activate	binary format
	= TV:	Timer Value:	integer format
Outputs:	– Q	Output	binary format
	= t:	actual value	integer format

Off-delay timer.

When In goes high, Q is immediately set high.

Q is set low at TV timer units after In was set low.

If In goes high-low-high for a period smaller than TV, then Q is unaffected.

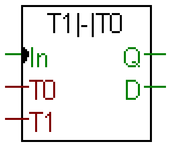
t indicates the value of the internal timer.

E.g.

```
In   ___---________--_---________
Q    ___-------____----------____
           <TV>         <TV>
```

4.4.4.5 On/off delay

On/off delay



[_onoffd2]

Inputs:	> In:	Activate	binary format
	= T0:	Timer Value 0	integer format
	= T1:	Timer Value 1	integer format
Outputs:	– Q	Output	binary format
	– D	Delay active/inactive	binary format

On-off-delay timer.

Output Q is set high at T1 timer units after input In was set high.

Output Q is set low at T0 timer units after input In was set low.

Output "D" = H: delay active,
Output "D" = L: delay inactive.

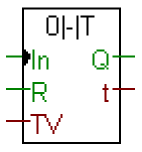
E.g.

```
In  ___-------_____________
  Q   _______-------------___
  <T1>     <---T0--->


  In  __-_------_-___________
  Q   ________--------------_
          <T1>    <---T0--->
```

4.4.4.6 Off delay with reset

Off delay with reset



[_offdelr]

Inputs:	> In:	Activate	binary format
	– R:	Reset	binary format
	= TV:	Timer Value:	integer format
Outputs:	– Q	Output	binary format
	= t:	actual value	integer format

Off-delay timer with reset.

When In goes high, Q is immediately set high.

Q is set low at TV timer units after In was set low.

If R is high the timer is cleared and Q is low.

If In goes high-low-high for a period smaller than TV, then Q is unaffected.

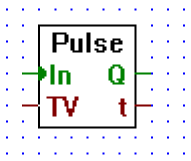
t indicates the value of the internal timer.

E.g.

```
In   ___---________--____________
R    ____________________------___
Q    ___-------____----__________
            <TV>      <TV>
```

4.4.4.7 Pulse

Pulse



[_pulse]

Inputs:	> In:	Activate	binary format
	= TV:	Timer Value:	integer format
Outputs:	– Q	Output	binary format
	= t:	actual value	integer format

Pulse timer, generates a pulse of duration TV when it detects a rising edge on the input.

If the input goes low before time TV, the pulse will be truncated.

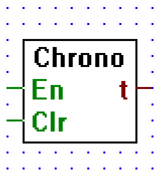
t indicates the value of the internal timer.

E.g.

```
In   __--------__--____
Q    __----______--____
       <TV>      <TV>
```

4.4.4.8 Chronometer

Chronometer

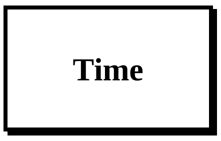


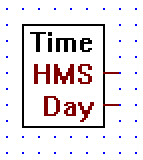
[_chrono2]

Inputs:	– En:	Enable	binary format
	– Clr:	Clear :	binary format
Outputs:	= t:	actual value	integer format

Calculates time intervals. The Clr input resets the timer. The En input starts the timer when high and stops it when low. The timer value is output at t.

4.4.4.9 Time (Hardware clock)





[_time]

Outputs:	= HMS:	Hours, Minutes, Seconds:	integer fornat
	= Day:	Day and Date	integer format

Reads the contents of the internal hardware clock into 2 integer outputs.

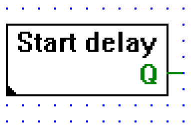
The outputs are set as follows:

HMS		Day	
Content	Digit nb.	Content	Digit nb.
0	9-6	0	9
Hours	5-4	Week	8-7
Minutes	3-2	Week Day	6
Seconds	1-0	Year	5-4
		Month	3-2
		Day	1-0

See also the description of the PCD's RTIME instruction.

4.4.4.10 Start delay

Start delay



[_stdel]

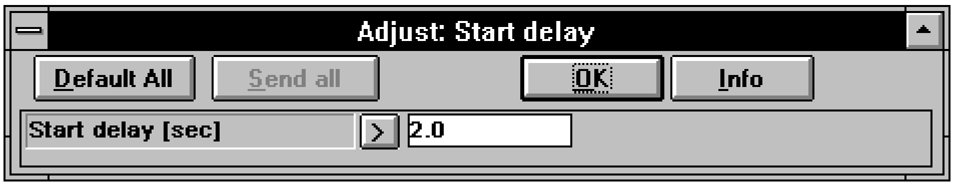
Output:	– Q:	Ready for start PCD program:	binary format

Start-delay timer.

Q is set low from the PCD's start up until the delay time is reached. Then it is kept high.

E.g.

```
PCD Start up _____----------------
Q            ______________-------
                  < Delay >
```

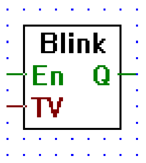


Start delay [sec] > 2.0

4.4.5 Blinker [sfupblin]

4.4.5.1 Blink delay T

Blink delay T



[_blink1]

Inputs:	– En:	Enable	binary format
	= TV:	Time Value	integer format
Output:	– Q:	Blinker output	binary format

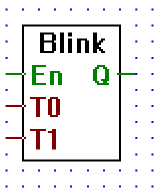
Blinks output Q with period TV while enable input En is high.

When En is low, Q is set low.

```
En   _____------------------____
Q    _____----_____----_____--____
          <TV>          <TV>
```

4.4.5.2 Blink delay T0/T1

Blink delay T0/T1



[_blink2]

Inputs:	– En:	Enable	binary format
	= T0	Time Value 0	integer format
	= T1	Time Value 1	integer format
Output:	– Q:	Blinker output	binary format

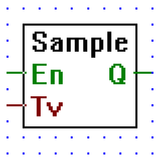
Blinks output Q with period T1 in the high state and a period T0 in the low state while enable input En is high.

When En is low, Q is set low.

```
En   _____-----------------------____
Q    _____----_______----_______--____
          <T1><-T0-><T1><-T0-><T1>
```

4.4.5.3 Sample

Sample



[_blinksamp]

Inputs:	– En:	Enable	binary format
	= Tv	Time value	integer format
Output:	– Q:	Sample output	binary format

When the signal enable En is high, Sample output Q with period Tv, in the high state during one program cycle, and output it in low state during the rest of the period.

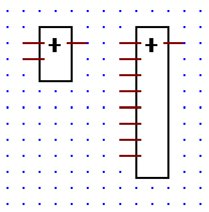
When En is low, Q is set low, and the timer is reset to 0.

```
En   ____-----------------___
Q    ____-____-___-____-____-_____
          <Tv>  > < Program Cycle
```

4.4.6 Integer arithmetic [sfupinte]

4.4.6.1 Add

Add



[_iadd]

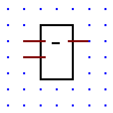
Inputs/Outputs: = all → integer format.

Adds the input values and transfers the result to the output.

Stretchable from 2 to 8 inputs.

4.4.6.2 Subtract

Subtract



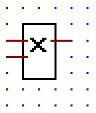
[_isub]

Inputs/Outputs: = all → integer format.

Outputs The result of the upper input minus the lower.

4.4.6.3 Multiply

Multiply



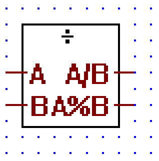
[_imul]

Inputs/Outputs: = all → integer format.

Outputs the result of the multiplication of its two inputs.

4.4.6.4 Divide

Divide



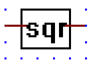
[_idiv]

Inputs/Outputs: = all → integer format.

Outputs A/B is set to the integer result of input A divided by input B. The remainder (modulo) is output to A%B.

4.4.6.5 Square root

Square root



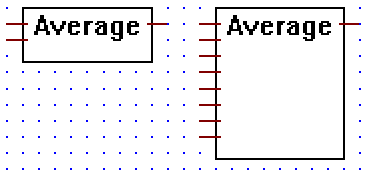
[_isqr]

Inputs/Outputs: = all → integer format.

Outputs the integer square root of its input.

4.4.6.6 Average

Average



[_iaverage]

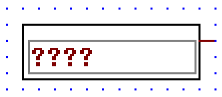
Inputs/Outputs: = all → integer format.

Ouputs the average of its input values.

Stretchable from 2 to 8 inputs.

4.4.6.7 Constant

Constant



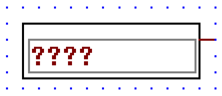
[_iconst]

Inputs/Outputs: = all → integer format.

Outputs the integer constant written in the entry-field.

4.4.6.8 Absolute

Absolute

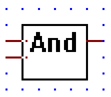


[_iabs]

Outputs the absolute value of the input.

4.4.6.9 Bitwise and

Bitwise and



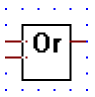
[_iand]

Inputs/Outputs: = all → integer format.

Outputs the logical AND of the first input with the second input.

4.4.6.10 Bitwise or

Bitwise or



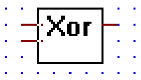
[_ior]

Inputs/Outputs: = all → integer format.

Outputs the logical OR of the first input with the second input.

4.4.6.11 Bitwise exclusive or

Bitwise exclusive or



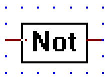
[_ixor]

Inputs/Outputs: = all → integer format.

Outputs the logical XOR of the first input with the second input.

4.4.6.12 Bitwise invert

Bitwise invert



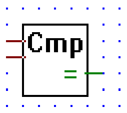
[_inot]

Inputs/Outputs: = all → integer format.

Outputs the invertion (1's complement) of the input.

4.4.6.13 Is equal to

Is equal to



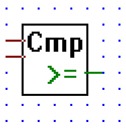
[_icmpeq]

Inputs: = all → integer format
Output: – binary format

Outputs binary high when both inputs are equal, else outputs binary low.

4.4.6.14 Is greater or equal to

Is greater or equal to



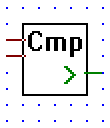
[_icmpge]

Inputs: = all → integer format
Output: – binary format

Outputs binary high when the upper input is greater or equal to the lower input, else outputs binary low.

4.4.6.15 Is greater than

Is greater than



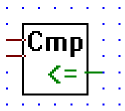
[_icmpgt]

Inputs: = all → integer format
Output: – binary format

Outputs binary high when the upper input is greater than the lower input, else outputs binary low.

4.4.6.16 Is smaller or equal to

Is smaller or equal to



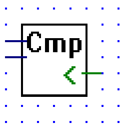
[_icmpse]

Inputs: = all → integer format
Output: – binary format

Outputs binary high when the upper input is less than or equal to the lower input, else outputs binary low.

4.4.6.17 Is smaller than

Is smaller than



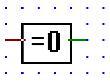
[_icmpst]

Inputs: = all → integer format
Output: – binary format

Outputs binary high when the upper input is less than the lower input, else outputs binary low.

4.4.6.18 Is zero

Is zero



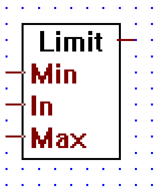
[_iiszero]

Inputs: = all → integer format
Output: – binary format

Outputs high when its integer input is zero, else output low.

4.4.6.19 Limit

Limit



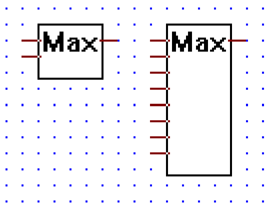
[_ilimit]

Inputs/Outputs: = all → integer format.

Outputs the input value In. The output value is limited between Max and Min.

4.4.6.20 Maximum

Maximum



[_imax]

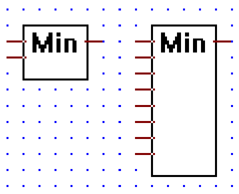
Inputs/Outputs: = all → integer format.

Outputs the largest of its input value.

Stretchable from 2 to 8 inputs.

4.4.6.21 Minimum

Minimum



[_imin]

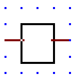
Inputs/Outputs: = all → integer format.

Outputs the smallest input value.

Stretchable from 2 to 8 inputs.

4.4.6.22 Move

Move



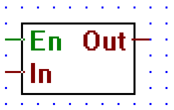
[_imove]

Inputs/Outputs: = all → integer format.

Outputs its integer input, used to connect an input label directly to an output label.

4.4.6.23 Move when enabled

Move when enabled



[_imovee]

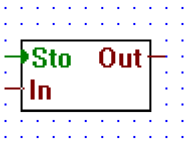
Inputs:	– En:	Enable (binary format)
	= In:	Input (integer format)
Output:	= Out	Output (integer format)

Outputs its input when the enable line is high.

If "En" = L, the output value is zero.

4.4.6.24 Move and store

Move and store

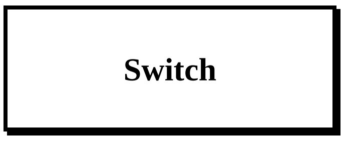


[_imoves]

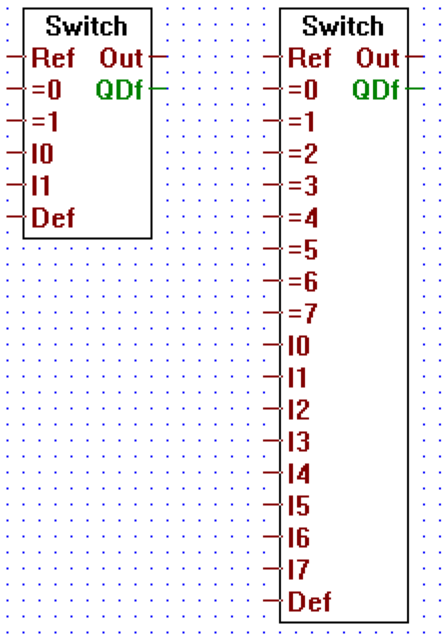
Inputs:	> Sto:	Store (Binary format)
	= In:	Input (integer format)
Output:	= Out:	Output (integer format)

Stores its input in an internal register when Sto goes from low to high.
Always outputs the contents of its internal register.

4.4.6.25 Switch



[_iswitch]



Inputs: = all: integer format

Outputs: = Out: integer format
– QDf binary format

When the input "Ref" is equal to one of its input "=n" it transfers the corresponding input "In" to the output.

When there are no input "=n" equal to "Ref" then the input "Def" is output to "Out", and the output QDf is set high.

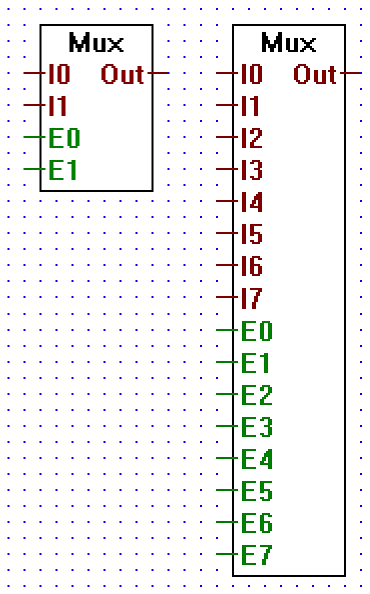
Stretchable from I1 to I7.

Input I0 as higher priority than I7. When more than one input "=n" equals the input "Ref", than only the input with the highest priority is transfered.

4.4.6.26 Multiplexer with binary selection

Mux bin selection

[_imux]



Inputs: = I0:
¦ integer format
= I7:
– E0:
¦ binary format
– E7:

Output: = Out: integer format

Outputs the state of input "In" when its corresponding enable input "En" is high.

Stretchable from 2 to 8 inputs "In".

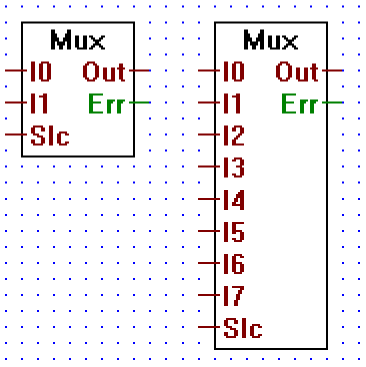
If all enable inputs "En" are low, then 0 is output.

The enable line E0 has the highest priority and E7 has the lowest. Therefore if 2 or more enable lines are set actived at the same time, the input associated with the highest priority enable line will be output.

4.4.6.27 Multiplexer with integer selection

Mux int selection

[_imux2]



Inputs: = I0:
¦ integer format
= I7:
= Slc integer format

Output: = Out: integer format
– Err: binary format

Transfers the input "In" to "Out". The input "In" transfered is selected via the input "Slc" (e.g. when Slc == 0 transfers I0, when Slc == 5 transfers I5...).

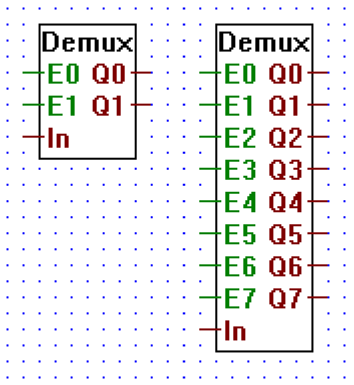
Stretchable from 2 to 8 inputs "In".

When "Slc" is out of range (i.e. "Slc" < 0 or "Slc" > n), "Out" outputs 0 and "Err" is set high.

4.4.6.28 Demultiplexer with binary selection

Demux bin selection

[_idemux]



Inputs:	– E0:	
	¦	binary format
	– E7:	
	= In:	integer format
Outputs:	= Q0:	
	¦	integer format
	= Q7:	

Transfers input In to an outputs Q0..Q7 when its corresponding enable input E0..E7 is high.

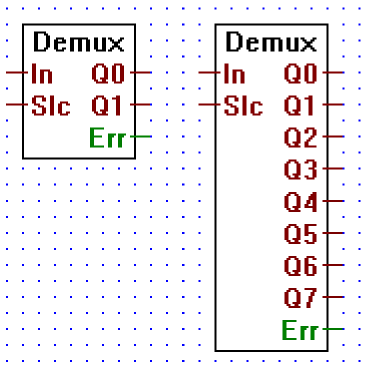
Stretchable from 2 to 8 inputs.

When an enable line is low, its corresponding output Q0..Q7 is set low.

4.4.6.29 Demultiplexer with integer selection

Demux int selection

[_idemux2]



Inputs:	=	In:	integer format
	=	Slc:	integer format
Outputs:	=	Q0	integer format
	¦		
	=	Q7	integer format
	–	Err:	binary format

Transfers its integer input "In" to a selected output (Q0..Qn). Outputs are selected via the input "Slc". E.g. when "Slc" == 1 the input "In" is transfered to "Q1".

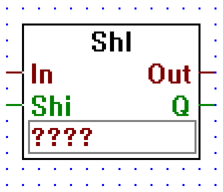
When not selected an output is set to 0.

Stretchable from 2 to 8 outputs.

Err goes high only when "Slc" is out of range, i.e. when "Slc" < 0 or when "Slc" > n.

4.4.6.30 Shift left

Shift left



[_ishftl2]

Inputs: = In: Input (integer format)
– Shi: Shift (binary format)

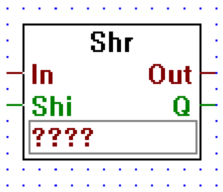
Outputs: = Out: Output (integer format)
– Q: Output (binary format)

Outputs to Out the integer input value from In shifted left by the number of bits indicated by the constant written in the edit-field.

The value of Q is set to the value of the last bit shifted out. The input Shi is the value to be shifted in to bit 0.

4.4.6.31 Shift right

Shift right



[_ishftr2]

Inputs: = In: Input (integer format)
– Shi: Shift (binary format)

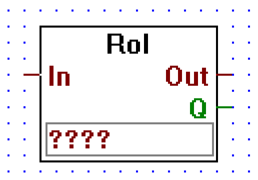
Outputs: = Out: Output (integer format)
– Q: Output (binary format)

Outputs to Out the integer input value from In shifted right by the number of bits indicated by the constant written in the edit-field.

The value of Q is set to the value of the last bit shifted out. The input Shi is the value to be shifted in to bit 31.

4.4.6.32 Rotate left

Rotate left



[_irotlf2]

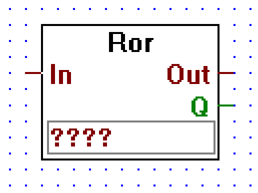
Inputs: = In: Input (integer format)

Outputs: = Out: Output (integer format)
– Q: Output (binary format)

Outputs to Out the integer input value from In rotated left by the number of bits indicated by the constant written in the edit-field.The value of Q is set to the value of the last bit rotated.

4.4.6.33 Rotate right

Rotate right



[_irotri2]

Inputs: = In: Input (integer format)

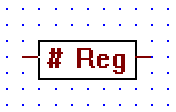
Outputs: = Out: Output (integer format)
– Q: Output (binary format)

Outputs to Out the integer input value from In rotated right by the number of bits indicated by the constant written in the edit-field.

The value of Q is set to the value of the last bit rotated.

4.4.6.34 Register indirect

Register indirect



[_iregin]

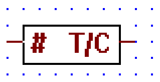
Inputs/Outputs: = all → integer format.

Outputs the value in register #.

E.g. if the # input is 5, then the value in register 5 is output.

4.4.6.35 T/C indirect

T/C indirect



[_itcind]

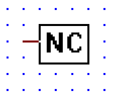
Inputs/Outputs: = all → integer format.

Outputs the value of timer/counter #.

E.g if the # input is 5, then the value in timer/counter 5 is output.

4.4.6.36 Not connected

Not connected



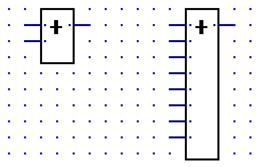
[_inotcn]

The "Not Connected" box, terminates unused integer outputs.

4.4.7 Floating point arithmetic [sfupfloa]

4.4.7.1 Add

Add



[_fadd2]

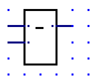
Inputs/Outputs: – all → floating point format

Adds the input values and transfers the result to the output.

Stretchable from 2 to 8 inputs.

4.4.7.2 Subtract

Subtract



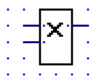
[_fsub]

Inputs/Outputs: – all → floating point format

Outputs the result of the upper input minus the lower.

4.4.7.3 Multiply

Multiply



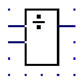
[_fmul]

Inputs/Outputs: – all → floating point format

Outputs the result of the multiplication of its two inputs .

4.4.7.4 Divide

Divide



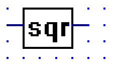
[_fdiv]

Inputs/Outputs: – all → floating point format

Outputs the result of the upper input divided by the lower

4.4.7.5 Square root

Square root



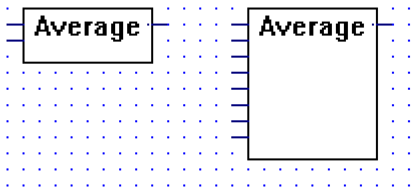
[_fsqr]

Inputs/Outputs: – all → floating point format

Outputs the square root of its input.

4.4.7.6 Average

Average



[_faverage]

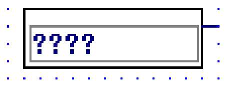
Inputs/Outputs: – all → floating point format

Ouputs the average of its input values.

Stretchable from 2 to 8 inputs.

4.4.7.7 Constant

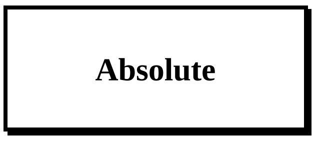
Constant

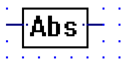


[_fconst]

Outputs the floating-point constant written in the entry-field.

4.4.7.8 Absolute



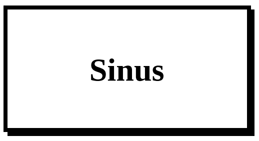


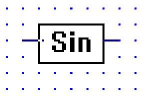
[_fabs]

Inputs/Outputs: – all → floating point format

Outputs the absolute value of the input.

4.4.7.9 Sinus



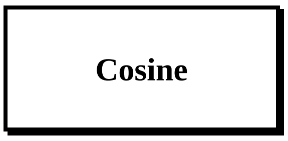


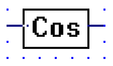
[_fsin]

Inputs/Outputs: – all → floating point format

Outputs the sine of ist input. The input is assumed to be in radians

4.4.7.10 Cosine





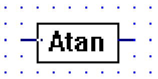
[_fcos]

Inputs/Outputs: – all → floating point format

Outputs the cosine of its input. The input is assumed to be in radians.

4.4.7.11 ARC tangent

ARC tangent



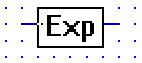
[_fatan]

Inputs/Outputs: – all → floating point format

Outputs the arc tangent of its input. The output is in radians.

4.4.7.12 Natural exponent

Natural exponent



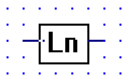
[_fexp]

Inputs/Outputs: – all → floating point format

Outputs 'e' to the power of the input.

4.4.7.13 Natural log

Natural log



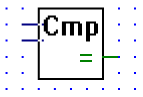
[_fln]

Inputs/Outputs: – all → floating point format

Outputs the natural logarithm of the input.

4.4.7.14 Is equal to

Is equal to



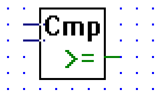
[_fcmpeq]

Inputs: all $\rightarrow$ floating point format
Output: binary format

Outputs binary high when both inputs are equal, else outputs binary low

4.4.7.15 Is greater or equal to

Is greater or equal to



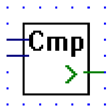
[_fcmpge]

Inputs: all $\rightarrow$ floating point format
Output: binary format

Outputs binary high when the upper input is greater or equal to the lower input, else outputs binary low.

4.4.7.16 Is greater than

Is greater than



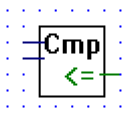
[_fcmpgt]

Inputs: all $\rightarrow$ floating point format
Output: binary format

Outputs binary high when the upper input is greater than the lower input, else outputs binary low.

4.4.7.17 Is smaller or equal to

Is smaller or equal to



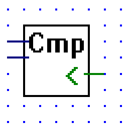
[_fcmpse]

Inputs: all → floating point format
Output: binary format

Outputs binary high when the upper input is less than or equal to the lower input, else outputs binary low.

4.4.7.18 Is smaller than

Is smaller than



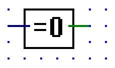
[_fcmpst]

Inputs: all → floating point format
Output: binary format

Outputs binary high when the upper input is less than the lower input, else outputs binary low.

4.4.7.19 Is zero

Is zero



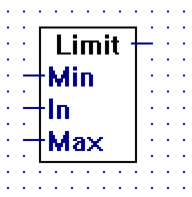
[_fiszero]

Input: floating point format
Output: binary format

Outputs high when its floating-point input is zero, else output low.

4.4.7.20 Limit

Limit



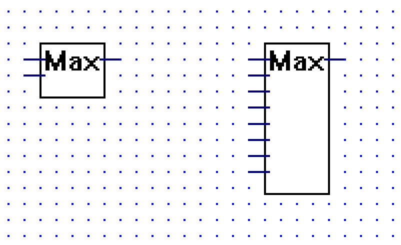
[_flimit]

Inputs/Outputs: – all → floating point format

Outputs the input value In. The output value is limited between Max and Min.

4.4.7.21 Maximum

Maximum



[_fmax2]

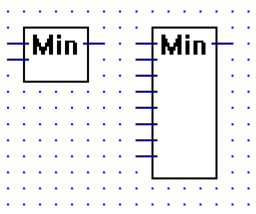
Inputs/Outputs: – all → floating point format

Outputs the largest input value.

Stretchable from 2 to 8 inputs.

4.4.7.22 Minimum

Minimum



[_fmin2]

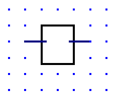
Inputs/Outputs: – all → floating point format

Outputs the smallest input value.

Stretchable from 2 to 8 inputs.

4.4.7.23 Move

Move



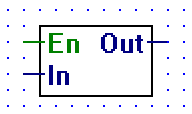
[_fmove]

Inputs/Outputs: – all → floating point format

Outputs its floating point input, used to connect an input label directly to an output label.

4.4.7.24 Move when enabled

Move when enabled



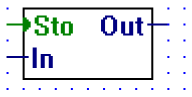
[_fmovee]

Inputs:	– En:	Enable (binary format)
	– In:	Input (floating point format)
Output:	– Out:	Output (floating point format)

Outputs its input when the enable input is high. If "En" = L, the output value is zero.

4.4.7.25 Move and store

Move and store

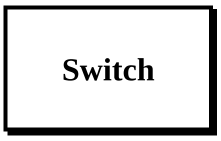


[_fmoves]

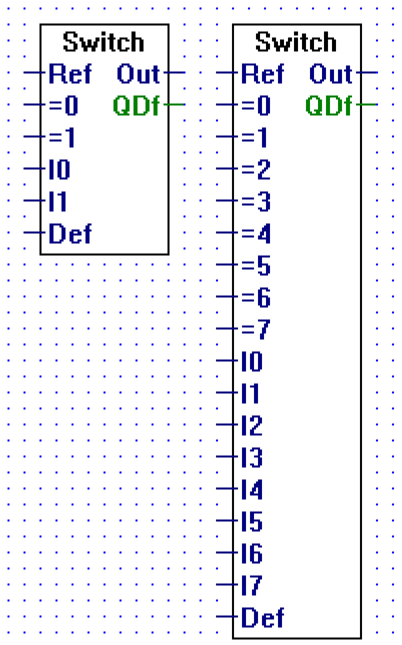
Inputs:	> Sto	Store (binary format)
	– In	Input (floating point format)
Output:	– Out	Output (floating point format)

Stores its input in an internal register when Sto goes from low to high. Always outputs the contents of its internal register.

4.4.7.26 Switch



[_fswitch]



Inputs: – all: → floating point format

Outputs: – Out: → floating point format
– QDf: → binary value

When the input "Ref" is equal to one of its input "=n" it transfers the corresponding input "In" to the output.

When there are no input "=n" equal to "Ref" then the input "Def" is output to "Out" and the output "QDf" is set high.

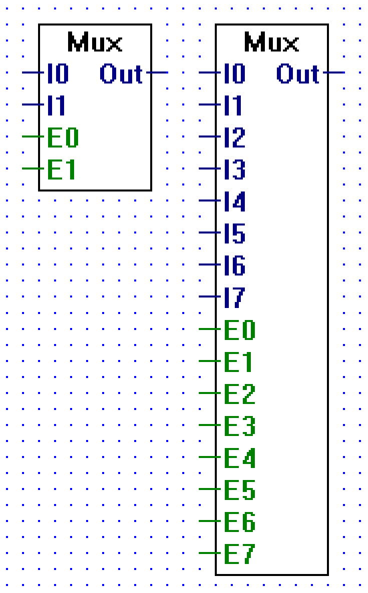
Stretchable from I1 to I7.

Input I0 as higher priority than I7. When more than one input "=n" equals the input "Ref", than only the input with the highest priority is transfered.

4.4.7.27 Multiplexer with binary selection

Mux with bin selection

[_fmux]



Inputs: – I0:
¦ floating point format
– I7:
– E0:
¦ binary format
– E7:

Output: – Out: floating point format

Transfers an input I0..I7 to the output when the corresponding enable signal E0..E1 is high.

Stretchable from 2 to 8 inputs.

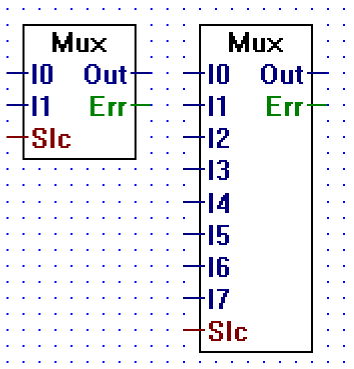
If all enable inputs are low, a low is output.

The enable line E0 has the highest priority and E7 has the lowest. Therefore if one or more enable lines are activated at the same time, the input associated with the highest priority enable line will be output.

4.4.7.28 Multiplexer with integer selection

Mux with int selection

[_fmux2]



Inputs:	– I0:	
	¦	floating point format
	– I7:	
	= Slc:	integer format
Outputs:	– Out:	floating point format
	– Err:	binay format

Transfers the input "In" to "Out". The input "In" transfered is selected via the input "Slc" (e.g. when Slc == 0 transfers I0, when Slc == 5 transfers I5...).

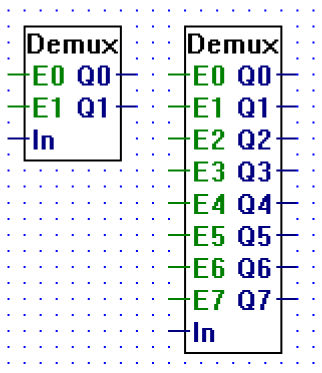
Stretchable from 2 to 8 inputs "In".

When "Slc" is out of range (i.e. "Slc" < 0 or "Slc" > n), "Out" outputs 0.0 and "Err" is set high.

4.4.7.29 Demultiplexer with binary selection

Demux with bin selection

[_fdemux]



Inputs: – E0:
¦ binary format
– E7:
= In: floating point format

Outputs: – Q0:
¦ floating point format
– Q7:

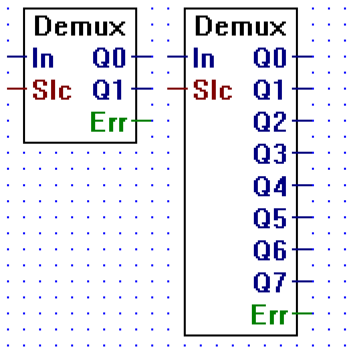
Transfers input In to output Qn when the corresponding enable line En is high.

Stretchable from 2 to 8.

When an enable line E0..E7 is low, its corresponding output Q0..Q7 is set low.

4.4.7.30 Demultiplexer with integer selection

Demux with int selection



[_fdemux2]

Inputs:	– In:	floating point format
	= Slc:	integer format
Outputs:	– Q0:	
	¦	floating point format
	– Q7:	
	– Err:	binary format

Transfers its integer input "In" to a selected output (Q0..Qn). Outputs are selected via the input "Slc". E.g. when "Slc" == 1 the input "In" is transfered to "Q1".

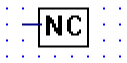
When not selected an output is set to 0.0.

Stretchable from 2 to 8 outputs.

Err goes high only when "Slc" is out of range, i.e. when "Slc" < 0 or when "Slc" > n.

4.4.7.31 Not connected

Not connected



[_Fnotcn]

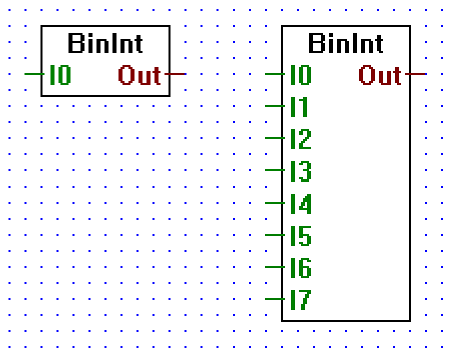
The "Not Connected" box, terminates unused floating point outputs.

4.4.8 Converters (binary-integer-floating point) [sfupconv]

4.4.8.1 Binary to integer 1-8 I/O/F

Bin to int 1-8

[_conbiti]



Inputs: – I0
¦ binary format
– I7

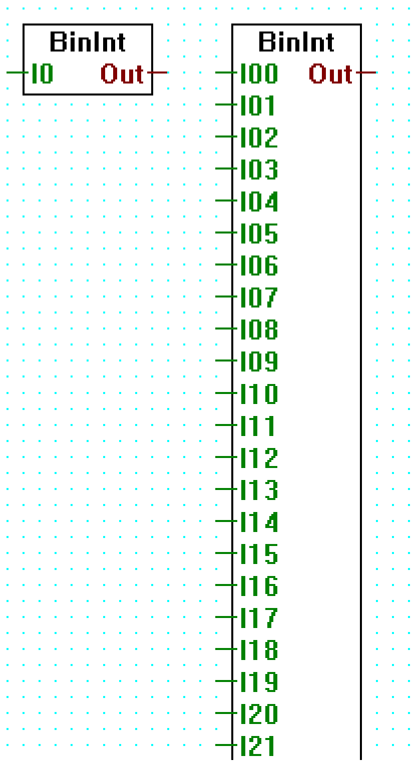
Output: = Out integer format

Transfers input I0..I7 to the bit 0..bit 7 of the output integer. All other bits of the output integer are cleared.Stretchable from 1 to 8 inputs.

4.4.8.2 Binary to integer 1-24 I/O/F

Bin to int 1-24

[_conbitim]



Inputs:	– I0:	
	¦	binary format
	– I23	
Output:	= Out	integer format

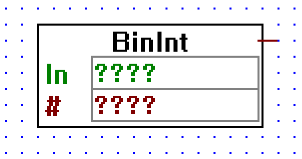
Transfers input I0..I23 to the bit 0..bit 23 of the output integer. All other bits of the output integer are cleared.

Stretchable from 1 to 23 inputs.

For more than 24 bits please use the Fbox 'Bin to int quick'.

4.4.8.3 Binary to integer quick

Bin to int quick



[_conbitiq]

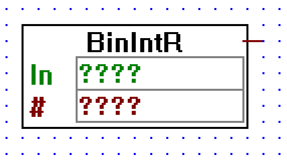
Output: = ⟶ integer format

Moves a sequence of bits into an integer. The first entry-field indicates the source address (I, O, F) of the binary sequence, and the second entry-field (#) indicates the number of bits to be moved.

The lowest addressed bit becomes the least significant bit in the destination integer. This is contrary to the PCA.

4.4.8.4 Binary to integer reverse quick

Bin to int reverse quick



[_conbitiqr]

Output: = ⟶ integer format

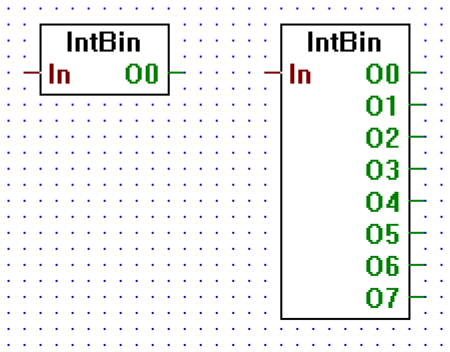
Moves a sequence of bits to an integer. The first entry-field indicates the source address (I, O, F) of the binary sequence, and the second entry-field (#) indicates the number of bits to be moved.

The highest addressed bit becomes the least significant bit in the destination integer. This is the same as for the PCA.

4.4.8.5 Integer to binary 1-8 O/F

Int to bin 1-8

[_conbito]



Input: = In: integer format

Outputs – O0:
¦ binary format
– O7:

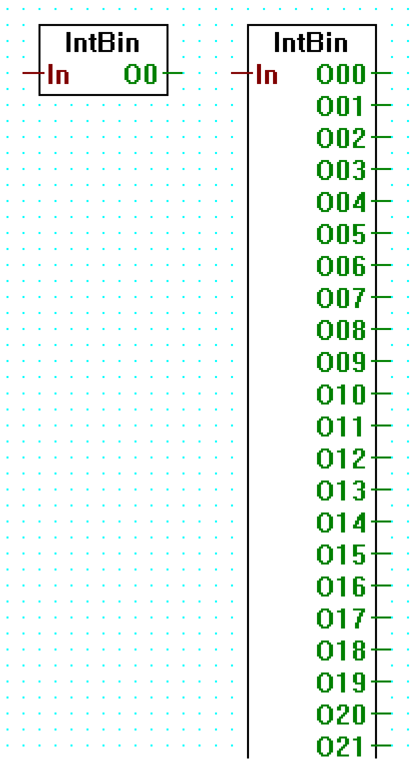
Transfers bits 0..7 of integer input In to binary outputs O0..7.

Stretchable from 1 to 8 bits.

4.4.8.6 Integer to binary 1-24 O/F

Int to bin 1-24

[_conbitom]



Input:	= In:	integer format
Outputs:	– O0	
		binary format
	– 023	

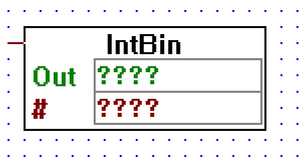
Transfers bits 0..23 of integer input In to binary outputs O0..23.

Stretchable from 1 to 24 bits.

For more than 24 bits please use the Fbox 'Int to bin quick'.

4.4.8.7 Integer to binary quick

Int to bin quick



[_conbitoq]

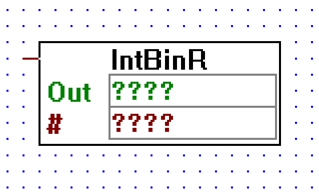
Input: = → integer format

Moves an integer value into a sequence of bits. The first entry-field indicates the destination address (O, F) of the binary sequence, and the second entry-field (#) indicates the number of bits to be moved.

The least significant bit of the incoming integer is moved to the lowest addressed bit. This is contrary to the PCA.

4.4.8.8 Integer to binary reverse quick

Int to bin reverse quick



[_conbitoqr]

Input: = → integer format

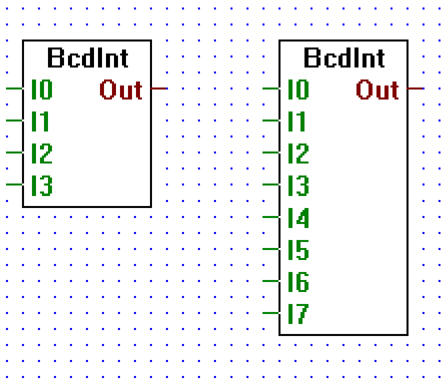
Moves an integer value into a sequence of bits. The first entry-field indicates the destination address (O, F) of the binary sequence, and the second entry-field (#) indicates the number of bits to be moved.

The least significant bit of the incoming integer is moved to the highest addressed bit. This is the same as the PCA.

4.4.8.9 BCD to integer

BCD to int

[_condigi]



Inputs: – I0:
¦ binary format
– I7:

Output: = Out: integer format

Reads a 4-bit BCD digit, and outputs it as an integer.

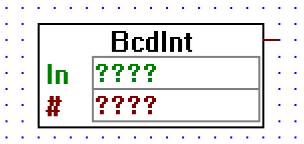
Stretchable, 4 or 8 inputs.

Input I0 is the least significant bit, I7 is the most significant.

4.4.8.10 BCD to integer quick

BCD to int quick

[_condigiq]



Output: = → integer format

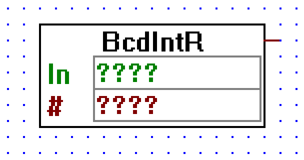
Reads Binary Coded Decimal (BCD) digits from Inputs, Outputs or Flags, and outputs them as an integer value.

The first operand is the address of the base address of the Inputs, Outputs or Flag, and the second operand indicates how many digits should be read.

The lowest addressed bit becomes the least significant bit of the least significant digit in the destination integer. This is contrary to the PCA.

4.4.8.11 BCD to integer reverse quick

BCD to int reverse quick



[_condigiqr]

Output: = → integer format

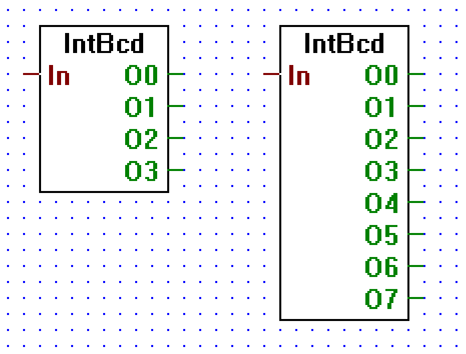
Help=Reads Binary Coded Decimal (BCD) digits from Inputs, Outputs or Flags, and outputs them as an integer value.

The first operand is the address of the base address of the Inputs, Outputs or Flag, and the second operand indicates how many digits should be read.

The lowest addressed bit is moved to the most significant bit of the integer output. This is the same as the PCA.

4.4.8.12 Integer to BCD

Int to BCD



[_condigo]

Input: = In: integer format

Outputs – O0:
¦ binary format
– O7:

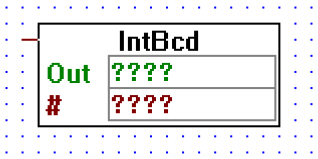
Converts integer value In into a 4 or 8-bit BCD value.

Stretchable, 4 or 8 outputs.

Output O0 is the least significant bit, O7 is the most significant.

4.4.8.13 Integer to BCD quick

Int to BCD quick



[_condigoq]

Input: = → integer format

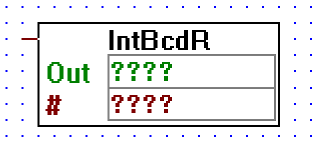
Help=Moves BCD digits from an integer to a sequence of Outputs or Flags.

The first operand is the address of the base address of the Outputs or Flag, and the second operand indicates how many digits should be moved.

The lowest addressed bit becomes the least significant bit of the least significant BCD digit. This is contrary to the PCA.

4.4.8.14 Integer to BCD reverse quick

Int to BCD reverse quick



[_condigoqr]

Input: = → integer format

Moves BCD digits from an integer to a sequence of Outputs or Flags.

The first operand is the address of the base address of the Outputs or Flag, and the second operand indicates how many digits should be moved.

The lowest integer bit becomes the highest addressed bit in the binary sequence. This is the same as the PCA.

4.4.8.15 1-bit to integer with shift

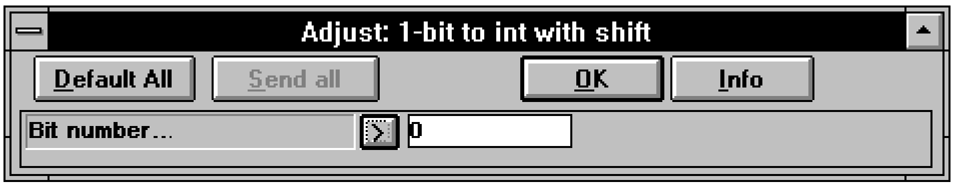
1-bit to int with shift

[_conbintoi]

Input:	–→	binary format
Output:	=→	integer format

Moves the input binary value in a specified bit of the output integer value. All other bits are set to 0.

The bit number chosen is specified by the "Bit number" variable in the adjust variable window.

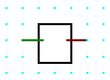

Adjust: 1-bit to int with shift

Default All | Send all | OK | Info

Bit number... 0

4.4.8.16 1-bit to integer LSB

1-bit to int LSB



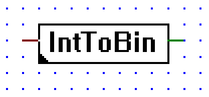
[_conbimove]

Input:	–→	binary format
Output:	=→	integer format

Outputs a 0 to the integer value when the input is LOW (0), else it outputs 1 to it.

4.4.8.17 Integer to 1-bit with shift

Int to 1-bit with shift

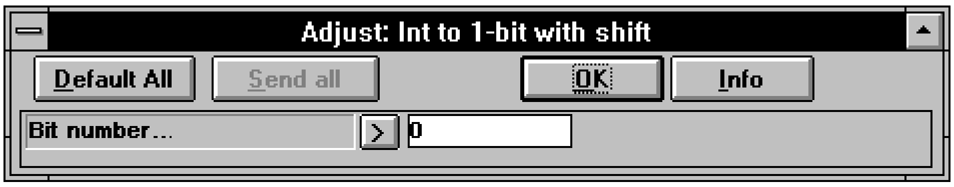


[_conitobin]

Input: – → integer format
Output: = → binary format

Moves a specified bit from the input integer value in the output binary value.

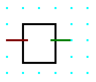
The bit number chosen is specified by the "Bit number" variable in the adjust variable window.



Bit number... 0

4.4.8.18 Integer LSB to 1-bit

Int LSB to 1-bit



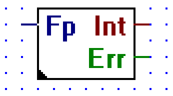
[_conibmove]

Input: =→ integer format
Output: – → binary format

Transfers LOW (0) in the binary output when its integer input is 0 (or any other even value), it transfers HIGH (1) when its integer input is 1 (or any other odd value).

4.4.8.19 Floating point format to integer format

Float to int



[_confpi2]

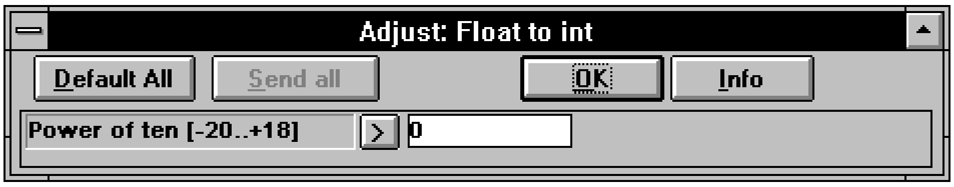
Input: – Fp: floating point format

Outputs: = Int integer format
– Err binary format

Outputs the integer representation of a floating point value.

The result is multiplied by 10 to the power of the adjustable parameter. E.g. if the input is 1234.56 and the entry field is -2, the integer result will be 12.

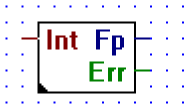
Err is set high if overflow occurs.



Power of ten [-20..+18] 0

4.4.8.20 Integer format to floating point format

Int to float



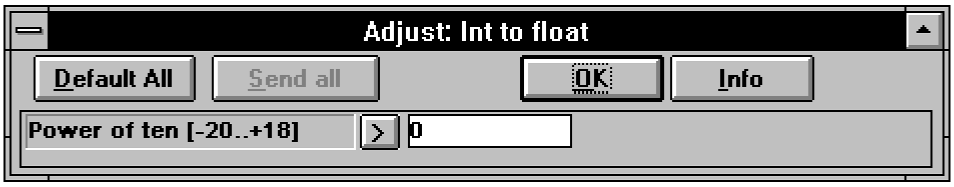
[_conifp2]

Input:	– Int:	integer format
Outputs:	= Fp	floating point format
	– Err	binary format

Outputs the floating point representation of an integer value.

The adjustable parameter is the power of 10 to which the integer value is to be raised. E.g. if the entry-field is 3 and the input is 12, then the result will be 12000.00.

Err is set high if overflow occurs.



Power of ten [-20..+18] 0

Notes :

4.4.9 Indirect addressing [sfupindi]

This function family is used for the indirect addressing of individual resources and sequences of resources. The main applications are:

- repetitive functions with different addresses
- selection of parameters in large tables
- wherever flexibility is wanted in addressing

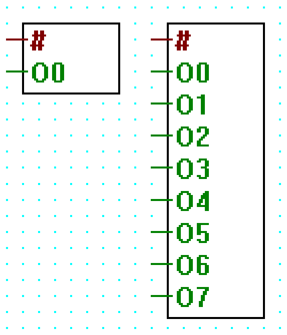
Although this function family can be used in a very universal way, it requires quite disciplined programming, as careless handling can result in addressing errors which are hard to locate. The following are the main areas for caution:

- Incorrect value at input #, especially when copying.
- Addressing errors when addresses are entered externally (process control system, terminal, BCD input). This can be prevented with a preceding "Limit" Fbox and a combined signal.
- Overlap with the dynamic address ranges.
- Timer / Counter distribution (timers become counters)
- I/O equipment.

4.4.9.1 Copy to outputs

Copy to outputs

[_indioo]



Inputs:	= #:	address of first element (0 - 8191)
	– O0	binary format
	¦	
	– O7	binary format

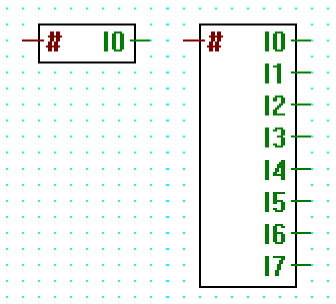
Copy 1 to 8 variables to consecutive outputs. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.2 Read from inputs

Read from inputs

[_indioi]



Input:	= #	address of first element (0 - 8191)
Ausgänge:	– I0	binary format
	¦	
	– I7	binary format

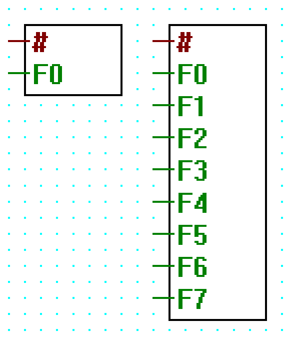
Read the content of 1 to 8 consecutive inputs or outputs. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.3 Copy to flags

Copy to flags

[_indflgo]



Inputs:	= #	address of first element (0 - 8191)
	– F0	binary format
	¦	
	– F7	binary format

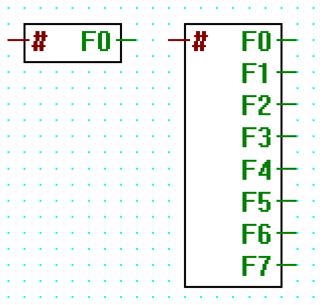
Copy 1 to 8 variables to consecutive flags. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.4 Read from flags

Read from flags

[_indflgi]



Input:	= #	address of first element (0 - 8191)
Outputs:	– F0	binary format
	¦	
	– F7	binary fomrat

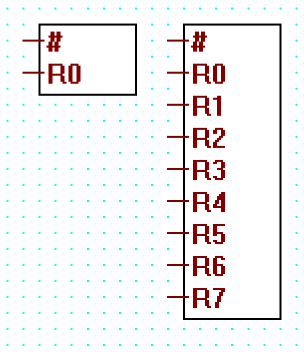
Read the content of 1 to 8 consecutive flags. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.5 Copy to registers integer

Copy to reg. integer

[_indrego]



Inputs:	= #	address of first element (0 - 4095)
	= R0	integer format
	¦	
	= R7	integer format

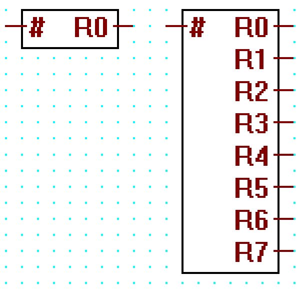
Copy 1 to 8 variables to consecutive registers. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.6 Read from registers integer

Read from reg. integer

[_indregi]



Input:	= #	address of first element (0 - 4095)
Outputs:	= R0	integer format
	¦	
	= R7	integer format

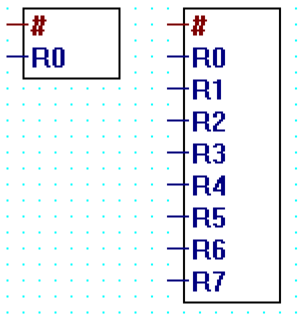
Read the content of 1 to 8 consecutive registers. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.7 Copy to registers float

Copy to reg. float

[_indfpo]



Inputs:	= #	address of first element (0 - 4095)
	= R0	floating point format
	¦	
	= R7	floating point format

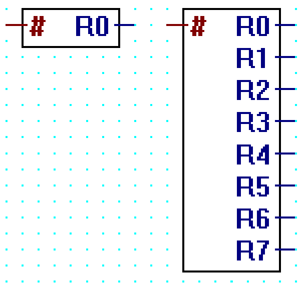
Copy 1 to 8 variables to consecutive floating point registers. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.8 Read from registers float

Read from reg. float

[_indfpi]



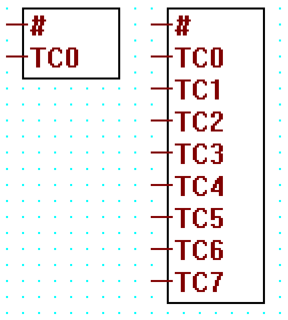
Input:	= #	address of first element (0 - 4095)
Outputs:	= R0	floating point format
	¦	
	= R7	floating point format

Read the content of 1 to 8 consecutive floating-point registers. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.9 Copy toTimer/Counter

[_indtco]



Inputs:	= #	address of first element (0 - 1599)
	= TC0	integer format
	¦	
	= TC7	integer format

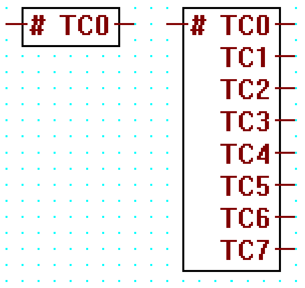
Copy 1 to 8 variables to consecutive timers or counters. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.10 Read from Timer/Counter

Read from T/C

[_indtci]



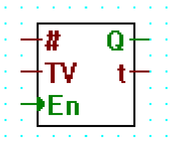
Input:	= #	address of first element (0 - 1599)
Outputs:	= TC0	integer format
	¦	
	= TC7	integer format

Read the content of 1 to 8 consecutive timers or counters. The address of the first element is indirectly given with the input #.

Please read the information's on the beginning of the chapter 4.4.9.

4.4.9.11 Timer with indirect addressing.

Timer



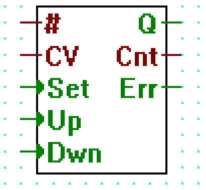
[_indtmr]

Inputs:	= #	address of first timer	
	= TV	Timer Value in 1/10 sec.	integer format
	> En	Enable (start of timer)	binary format
Outputs:	– Q	timer output	binary format
	= t	actual timer stand	integer format

Timer function with indirect address. The input # gives the address of the timer used.

4.4.9.12 Counter with indirect addressing

Counter



[_indcnt]

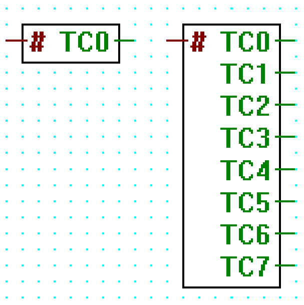
Inputs:	= #	address of first counter	
	= CV	load value	integer format
	> Set	load with "CV"	binary format
	> Up	increment (+1)	binary format
	> Dwn	decrement (–1)	binary format
Outputs:	– Q	Output (H, if C > 0)	binary format
	= Cnt	actual counter stand	integer format
	– Err	Error (overflow)	binary format

Counter function with indirect address. The input # gives the address of the used counter.

4.4.9.13 Read logic state from Timer/Counter

Read logic state T/C

[_indtcli]



Input:	= #	first T/C address (0 - 1599)	
Outputs:	– TC0	H/L Timer/Counter	binary format
	¦		
	– TC7	H/L Timer/Counter	binary format

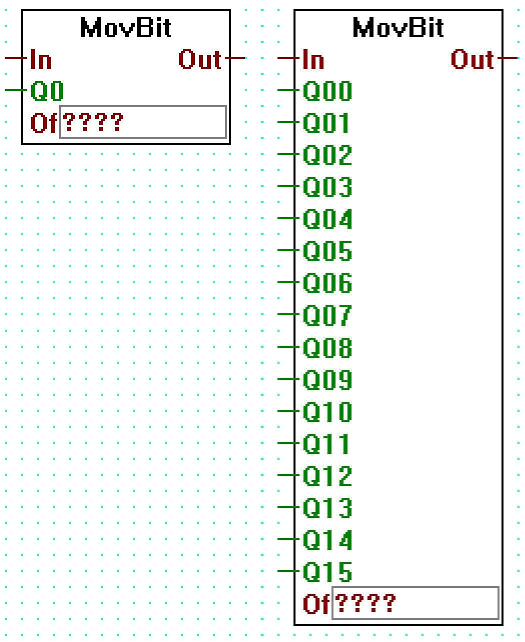
Read the logical state of 1 to 8 consecutive timers or counters. The address of the first element is indirectly given with the input #.

4.4.10 Move Data [sfupmove]

4.4.10.1 Move-In Bit

Move-In Bit

[_movebiti]



Inputs:	= In:	integer format
	– Q00:	binary format (Q = Quantum) (0 or 1)
	¦	
	– Q15:	binary format
	= Of:	Offset, integer format (0 -31)
Output:	= Out:	integer format

Move 1 to 24 bits in a register.

The input In is combined with the bits from inputs Q0 to Q.. (last used). The bits Q0 is moved to the location defined by the offset value Of.

The next bits used are moved in the successive locations.

All other bits are not changed. The result is copied to the output Out.

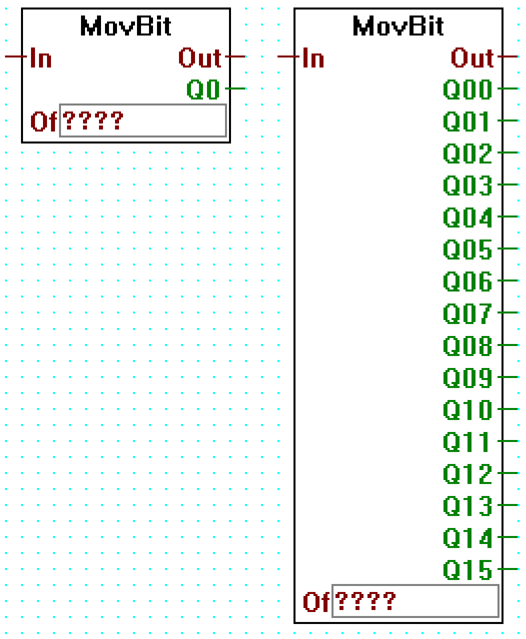
E.g. "Move-In Bit" with "Of" = 5 and inputs Q0 to Q7:

```
                    Q7         Q0
                    x xxxx xxx
                    ↓ ↓↓↓↓ ↓↓↓
oooo oooo oooo oooo ooox xxxx xxxo oooo
31                              5     0
```

4.4.10.2 Move-Out Bit

Move-Out Bit

[_movebito]



Inputs:	= In:	integer format
	= Of:	Offset, integer format 0 -31)
Outputs:	= Out:	integer format
	– Q00:	binary format (Q = Quantum) (0 or 1)
	¦	
	– Q15:	binary format

Bits are moved out from the value of the input In. The first bit B0 is moved from the location defined by the offset value Of.

The next bits used are moved from successive locations.

The input In is unchanged and copied to the output Out.

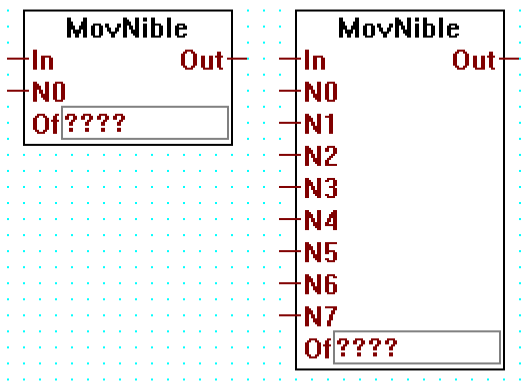
E.g. "Move-Out Bit", "Of" = 5 and outputs Q0 to Q7:

```
31                            5     0
 oooo oooo oooo oooo ooox xxxx xxxo oooo
                        ↓ ↓↓↓↓ ↓↓↓
                        x xxxx xxx
                       Q7        Q0
```

4.4.10.3 Move-In Nibble (4 Bit binär)

Move-In Nibble

[_movenibi]



Inputs:	= In:	integer format
	= N0:	integer format (0 - 15)
	¦	
	= N7:	integer format
	= Of:	Offset, integer format (0 - 7)
Output:	= Out:	integer format

Move 1 to 8 nibbles in a register.

The input In is combined with the nibbles (4 bits) from inputs N0 to N.. (last used). The LS-nibble from N0 is moved to the nibble location defined by the offset value Of.

The next nibbles used are moved in the successive locations.

All other nibbles are not changed. The result is copied to the output Out.

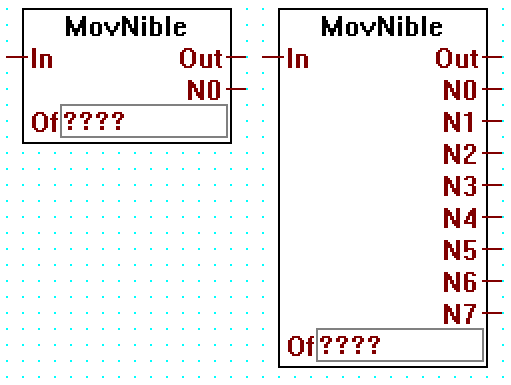
E.g. "Move-In Nibbles" with "Of" = 3 and inputs N0 to N2:

```
           N2   N1   N0
          xxxx xxxx xxxx
          ↓↓↓↓ ↓↓↓↓ ↓↓↓↓
 oooo oooo oooo oooo oooo oooo oooo oooo
Ni 7    6    5    4    3    2    1    0
```

4.4.10.4 Move-Out Nibble (4 Bit binär)

Move-Out Nibble

[_movenibo]



Inputs:	= In:	integer format
	= Of:	Offset, integer format (0 - 7)
Outputs:	= Out:	integer format
	= N0:	integer format (0 - 15)
	¦	
	= N7:	integer format

Nibbles are moved out from the value of the input In. The first nibble N0 is moved from the location defined by the offset value Of.

The next nibbles used are moved from successive locations.

The input In is unchanged and copied to the output Out.

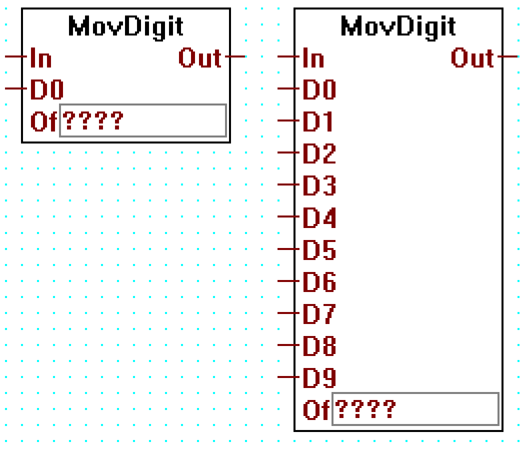
E.g. "Move-Out Nibble", "Of" = 3 and outputs N0 to N2:

```
Ni 7    6    5    4    3    2    1    0
 oooo oooo oooo oooo oooo oooo oooo oooo
           ↓↓↓↓ ↓↓↓↓ ↓↓↓↓
           xxxx xxxx xxxx
            N2   N1   N0
```

4.4.10.5 Move-In Digit (4 Bit BCD)

Move-In Digit

[_movedigi]



Inputs:	= In:	integer format
	= D0:	integer format (0 - 9)
	¦	
	= D9:	integer format
	= Of:	Offset, integer format (0 - 9)
Output:	= Out:	integer format

Move 1 to 10 digits in a register.

The input In is combined with the digit from inputs D0 to D.. (last used). The LS-digit from D0 is moved to the digit location defined by the offset value Of.

The next digits used are moved in the successive locations.

All other digits are not changed. The result is copied to the output Out.

E.g. "Move-In Digit" with "Of" = 3 and inputs D0 to D2:

```
                         D2  D1  D0
                          x   x   x
                          ↓   ↓   ↓
          o   o   o   o   x   x   x   o   o   o
Digit     9   8   7   6   5   4   3   2   1   0
```

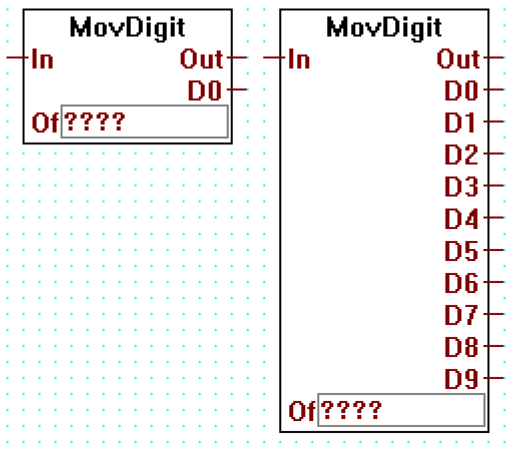
Only positive values can be treated.

The max. value for digit 9 is 2

4.4.10.6 Move-Out Digit (4 Bit BCD)

Move-Out Digit

[_movedigo]



Inputs:	= In:	integer format
	= Of:	Offset, integer format (0 - 9)
Outputs:	= Out:	integer format
	= D0:	integer format (0 - 9)
	¦	
	= D9:	integer format

Digits are moved out from the value of the input In. The first digit D0 is moved from the location defined by the offset value Of.

The next digits used are moved from successive locations.

The input In is unchanged and copied to the output Out.

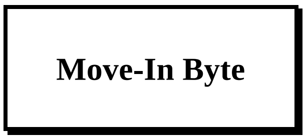
E.g. "Move-Out Digit", "Of" = 3 and outputs D0 to D2:

```
Digit   9   8   7   6   5   4   3   2   1   0
        o   o   o   o   x   x   x   o   o   o
                        ↓   ↓   ↓
                        x   x   x
                       D2  D1  D0
```

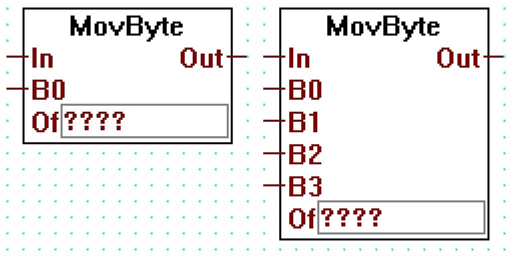
Only positive values can be treated.

The max. value for digit 9 is 2

4.4.10.7 Move-In Byte (8 Bit)



[_movebyti]



Inputs:	= In:	integer format
	= B0:	integer format (0 - 255)
	¦	
	= B3:	integer format
	= Of:	Offset, integer format (0 - 3)
Outputs:	= Out:	integer format

Move 1 to 4 bytes in a register.

The input In is combined with the bytes from inputs B0 to B.. (last used). The LS-byte from B0 is moved to the byte location defined by the offset value Of.

The next bytes used are moved in the successive locations.

All other bytes are not changed. The result is copied to the output Out.

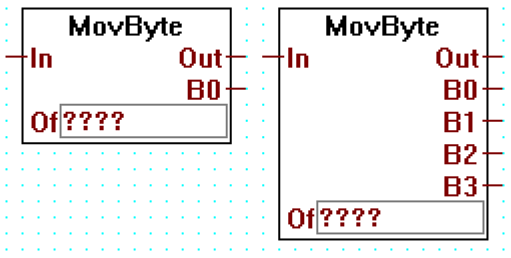
E.g. "Move-In Byte" with "Of" = 1 and inputs B0 and B1:

```
             B 1        B 0
          xxxx xxxx xxxx xxxx
          ↓↓↓↓ ↓↓↓↓ ↓↓↓↓ ↓↓↓↓
oooo oooo xxxx xxxx xxxx xxxx oooo oooo
 Byte 3    Byte 2    Byte 1    Byte 0
```

4.4.10.8 Move-Out Byte (8 Bit)

Move-Out Byte

[_movebyto]



Inputs:	= In:	integer format
	= Of:	Offset, integer format (0 - 3)
Outputs:	= Out:	integer format
	= B0:	integer format (0 - 255)
	¦	
	= B3:	integer format (0 - 255)

Bits are moved out from the value of the input In. The first bit B0 is moved from the location defined by the offset value Of.

The next bits used are moved from successive locations.

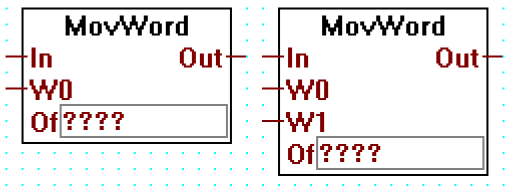
The input In is unchanged and copied to the output Out.

E.g. "Move-Out Byte", "Of" = 1 and outputs B0 and B1:

```
 Byte 3     Byte 2     Byte 1     Byte 0
oooo oooo xxxx xxxx xxxx xxxx oooo oooo
          ↓↓↓↓ ↓↓↓↓ ↓↓↓↓ ↓↓↓↓
          xxxx xxxx xxxx xxxx
             B 1       B 0
```

4.4.10.9 Move-In Word (16 Bit)

[_movewori]



Inputs:	= In:	integer format
	= W0:	integer format (0 - 65535)
	¦	
	= W1:	integer format
	= Of:	Offset, integer format (0 - 1)
Outputs:	= Out:	integer format

Move 1 to 2 words in a register.

The input In is combined with the words (16 bits) from inputs W0 to W.. (last used). The LS-word from W0 is moved to the word location defined by the offset value Of.

The next words used are moved in the successive locations.

All other words are not changed. The result is copied to the output Out.

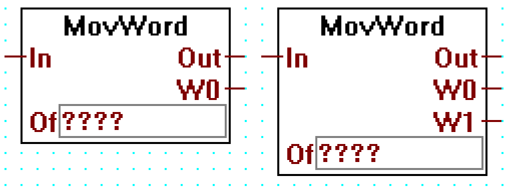
E.g. "Move-In Word" with "Of" = 1 and input W (0):

```
    Input W (0)
xxxx xxxx xxxx xxxx
↓↓↓↓ ↓↓↓↓ ↓↓↓↓ ↓↓↓↓
xxxx xxxx xxxx xxxx oooo oooo oooo oooo
        W 1                  W 0
```

4.4.10.10 Move-Out Word (16 Bit)

Move-Out Word

[_moveworo]



Inputs:	= In: integer format
	= Of: Offset, integer format (0 - 1)
Outputs:	= Out: integer format
	= W0: integer format (0 - 65535)
	¦
	= W1: integer format

Words are moved out from the value of the input In. The first word W0 is moved from the location defined by the offset value Of.

The next words used are moved from successive locations.

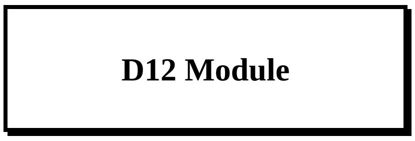
The input In is unchanged and copied to the output Out.

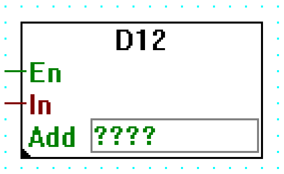
E.g. "Move-Out Word", "Of" = 1 and output W0:

```
        W 1                 W 0
xxxx xxxx xxxx xxxx oooo oooo oooo oooo
↓↓↓↓ ↓↓↓↓ ↓↓↓↓ ↓↓↓↓
xxxx xxxx xxxx xxxx
    Output W0
```

4.4.11 Displays [sfupdisp]

4.4.11.1 Display module PCA2.D12





[_dispd12]

Inputs:	– En:	Enable
	= In:	Integer value

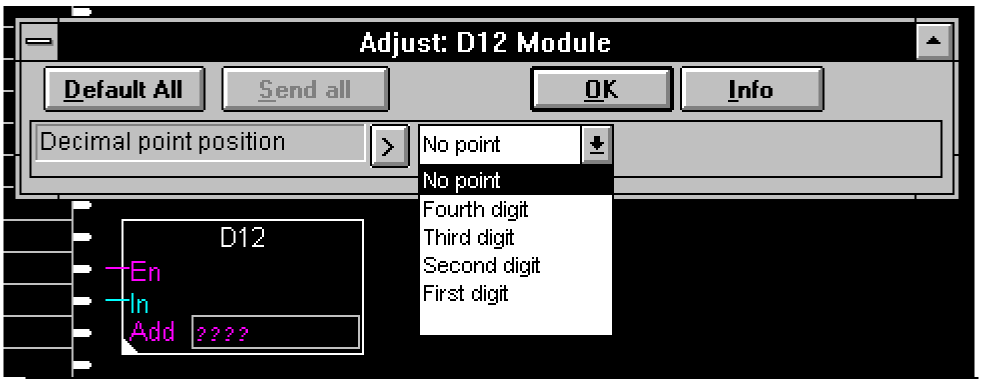
Outputs one integer value to a PCA2.D12 display module.

When the function box is enabled, the 4 least significant digits of the integer are shown on the display

The D12 is enabled with the En input. The Add input contains the base address of the D12 module. This function box assumes that Clk, D-IN and En of the D12 module are connected to outputs Add+0, Add+1 and Add+2 respectively.

The display remains unchanged if the function box is not enabled.

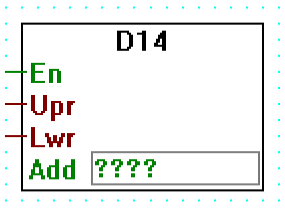
A decimal point can be adjusted manually inside the function box. In order to do so the user must double click on the function box when he is in selection mode. This parameter will be effective only after a recompilation of the program.



4.4.11.2 Display module PCA2.D14

D14 Module

[_dispd14]



Inputs: – En: Enable
= Upr: Upper display - Integer value
= Lwr: Lower display - Integer value

Outputs two integer values to a PCA2.D14 display module.

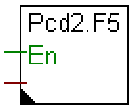
When the function box is enabled, the 6 least significant digits of the Upr integer are shown in the upper display, and the 6 least significant digits of the Lwr integer are shown in the lower display.

The D14 is enabled with the En input. The Add input contains the base address of the D14 module. This function box assumes that Clk, D-IN and En of the D14 module are connected to outputs Add+0, Add+1 and Add+2 respectively.

The display remains unchanged if the function box is not enabled.

4.4.11.3 Display module PCD2.F510 for numeric displays

PCD2.F510

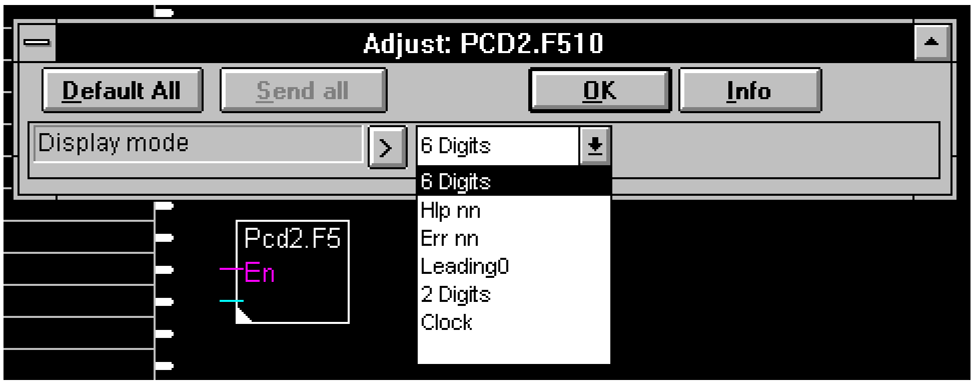


[_disppcd23]

Input: – Enable

Outputs to the PCD2.F5 display module in numeric format. PCD2.F5 can display up to 6 digits (-99999..999999).

It can also display predefined numeric expressions. The display format is selected in the adjust window. If input Enable is Low, the last defined format is valid. The default format is "6 Digits"



Display mode: 6 Digits

The different selection modes are:

Selection:	what is displayed:
● 6 Digits	6 digits from Val. If Val is not between -99999 and 999999 the display is cleared.
● Hlp nn	'HLP' with 2 digits from Val
● Err nn	'Err' with 2 digits from Val. If val is not between 0-99 the display is cleared.
● Leading 0	6 digits with leading zero
● 2 Digits	2 digits mode. If val is not between 0-99 the display is cleared.
● Clock	Display the Val using a clock format.

4.4.11.4 Display module PCD2.F510 for text displays

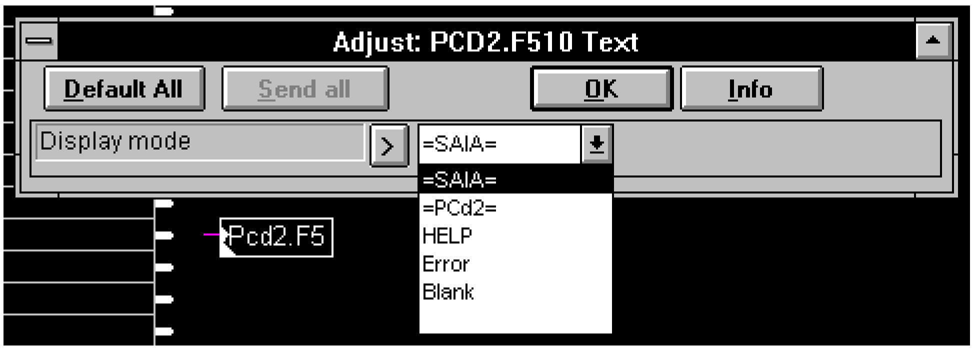
PCD2.F510 Text

[_disppcd24]

Input: – Enable

Outputs to the PCD2.F5 display module in text format.

It can also display predefined texts. The display format is selected in the adjust window. If input Enable is Low, the last defined format is valid. The default format is "=SAIA=".



Display mode =SAIA=

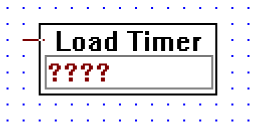
The different selection modes are:

Selection:	what is displayed:
• =SAIA=	'=SAIA='
• =PCd2=	'PCd2='
• HELP	' HELP '
• Error	' Error'
• Blank	The display will be cleared.

4.4.12 GRAFTEC functions [sfupgraf]

4.4.12.1 Load timer

Load timer



[_grafldt]

Input: = Load value: register or constant (integer format)

Fbox for loading and activating (starting) a timer in a step.

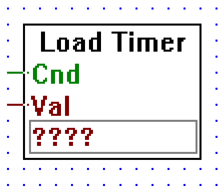
The entry field should be completed either with a symbol name or the absolute timer address. (Symbol names are preferable, as symbol management is carried out automatically).

The value to be loaded should be entered at the numeric input.

For ONLINE viewing of timer content, the symbol name or absolute timer address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.2 Load timer conditional

Load timer conditional



[_grafcldt]

Inputs: – Cnd: Condition H/L, (binary format)
= Val: Load value: register or constant (integer format)

Fbox for conditional loading and activating (starting) a timer in a step.

The entry field should be completed either with a symbol name or the absolute timer address. (Symbol names are preferable, as symbol management is carried out automatically).

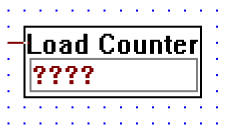
The value to be loaded should be entered at the numeric input.

Loading only takes place when the condition on the binary input "Cnd" is met.

For ONLINE viewing of timer content, the symbol name or absolute timer address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.3 Load counter

Load counter



[_grafldc]

Input: = Load value: register or constant (integer format)

Fbox for loading a counter in a step.

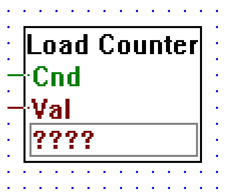
The entry field should be completed either with a symbol name or the absolute counter address. (Symbol names are preferable, as symbol management is carried out automatically).

The value to be loaded should be entered at the numeric input.

For ONLINE viewing of counter content, the symbol name or absolute counter address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.4 Load counter conditional

Load counter conditional



[_grafcldc]

Inputs: – Cnd: Condition H/L, (binary format)
= Val Load value: register or constant (integer format)

Fbox for conditional loading a counter in a step.

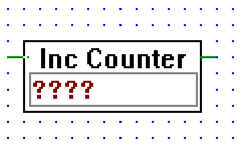
The entry field should be completed either with a symbol name or the absolute counter address. (Symbol names are preferable, as symbol management is carried out automatically).

The value to be loaded should be entered at the numeric input. Loading only takes place when the condition on the binary input "Cnd" is met.

For ONLINE viewing of counter content, the symbol name or absolute counter address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.5 Increment counter

Increment counter



[_grafincc]

Input: – Condition for increment H/L (binary format)
Output: – logical state of the counter H/L (binary format)

Fbox for incrementing a counter (+1), normally in a step.

The counter must first have been loaded using the function “Load Counter”.

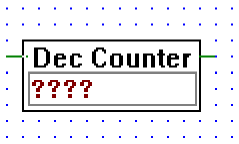
The corresponding name or counter address from the “Load Counter” instruction should be indicated in the entry field.

Incrementating only takes place when the input condition is met.

For ONLINE viewing of counter content, the symbol name or absolute counter address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.6 Decrement counter

Decrement counter



[_grafdecc]

Input: – Condition for decrement H/L (binary format)
Output: – logical state of the counter H/L (binary format)

Fbox for decrementing a counter (-1), normally in one step.

The counter must first have been loaded using the function “Load Counter”.

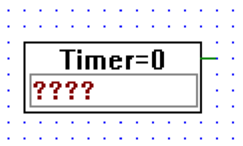
The corresponding name or counter address from the “Load Counter” instruction should be indicated in the entry field.

Decrementing only takes place when the input condition is met.

For ONLINE viewing of counter content, the symbol name or absolute counter address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.7 Timer is zero

Timer is zero



[_graftisl]

Output: – H if timer is 0. (binary format)

Fbox for querying the logical state of a timer in a TRANSITION. The function is true if the timer has reached zero. If this FBox stands alone in a TRANSITION, the next step switches in after the timer has reached zero.

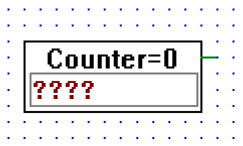
This function is linkable, preferably with Kontaktplan symbols.

The corresponding name or timer address from the “Load Timer” instruction should be indicated in the entry field.

For ONLINE viewing of timer content, the symbol name or absolute timer address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.8 Counter is zero

Counter is zero



[_grafcisl]

Output: – H if timer is 0. (binary format)

Fbox for querying the logical state of a counter in a TRANSITION. The function is true if the counter has reached zero.

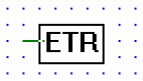
This function is linkable, preferably with Kontaktplan symbols.

The corresponding name or counter address from the “Load Timer” instruction should be indicated in the entry field.

For ONLINE viewing of counter content, the symbol name or absolute counter address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.9 End of transition

End of transition



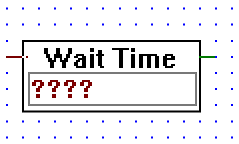
[_grafetrsv]

Input: – binary format

This fbox is used in a transition. When its input is high it indicates that the transition is completed else it is not.

4.4.12.10 Wait Time

Wait Time



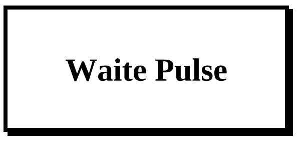
[_grafwait]

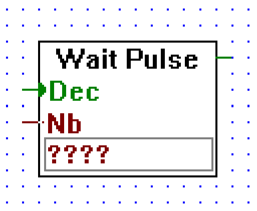
Input: = Time value: register or constant (integer format)
Output: – H if timer is 0. (binary format)

Compact Fbox to program a wait pause in a TRANSITION. The first time the TRANSITION is exectuted, the timer is started with the value at the input. The output is true if the timer has reached zero.

For ONLINE viewing of timer content, the symbol name or absolute timer address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.12.11 Waite Pulse





[_grafwpls]

Input: – Dec: Decrement (binary format)
= Nb: Load value: register or constant (integer format)

Output: – H if counter is 0. (binary format)

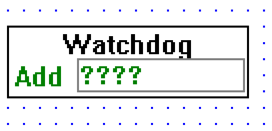
Compact Fbox to program a pause waiting pulses in a TRANSITION. The first time the TRANSITION is exectuted, the counter is started with the value at the input. Each pulse at the input Dec will decrement the counter by one. The output is true if the counter has reached zero.

For ONLINE viewing of counter content, the symbol name or absolute counter address should be entered in an entry field on the left-hand margin of the FUPLA screen. In ONLINE operation, it is possible to insert an online probe (online box) at this entry field.

4.4.13 Special functions [sfupspec]

4.4.13.1 Watch dog

Watch dog



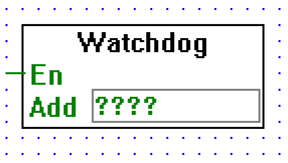
[_watchdog]

When this function box is called at a frequency ≥ 5Hz it keeps the watch-dog circuit active.

The element address "Add" is normally 255. (See hardware manuals PCD2, PCD4, PCD6).

4.4.13.2 Watch dog with enable

Watch dog enable



[_watchdoge]

Input: – En Enable (binary format)

When this function box is called with its entry En (enable) at a frequency ≥ 5Hz it keeps the watchdog circuit active.

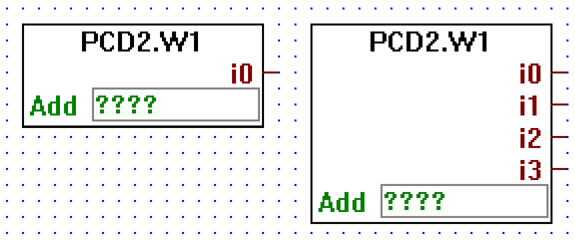
The element address "Add" is normally 255. (See hardware manuals PCD2, PCD4, PCD6).

Notes :

4.4.14 Analog modules [sfupanlg]

4.4.14.1 Analog Input module PCD2.W1 (12 bit)

[_anad2w1]



Inputs: = i0
¦ : Integer values
= i3

This function box outputs the A/D conversion of the PCD2.W1 analogic input card.

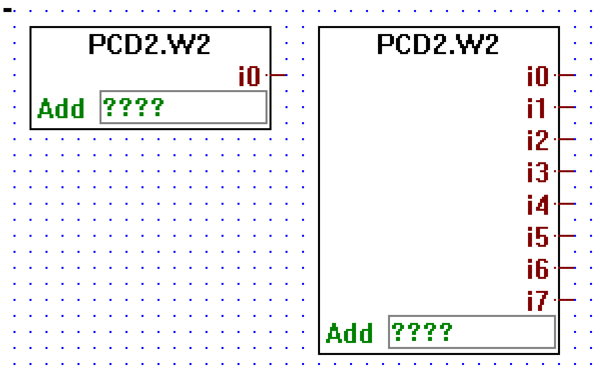
Each time this function box is executed one input channel is converted. When the conversion of a channel is not ready this function box outputs its last valid conversion. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD2.W1 must be used for each PCD2.W1 card used.

The parameter Add is the base address of the card. E.g.: O 16. Output 240 must never be used for analogic module.

4.4.14.2 Analog Input module PCD2.W2 (8 Bit)

[_anad2w2]



Inputs: = i0
¦ Integer values
= i7

This function box outputs the A/D conversion of the PCD2.W2 analogic input card.

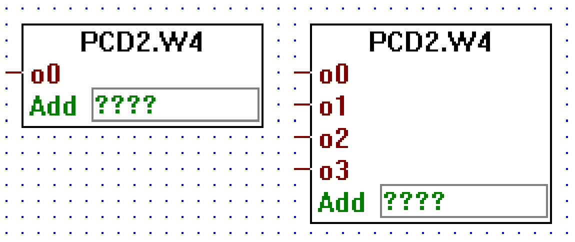
Each time this function box is executed one input channel is converted. When the conversion of a channel is not ready this function box outputs its last valid conversion. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD2.W2 must be used for each PCD2.W2 card used.

The parameter Add is the base address of the card. E.g.: O 16. Output 240 must never be used for analogic module.

4.4.14.3 Analog Output module PCD2.W4 (8 Bit)

[_anad2w4]



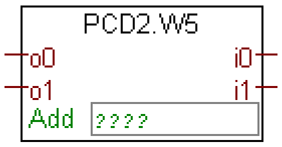
Outputs: = o0
¦ Integer values
= o3

This function box transfers its (1 to 4) inputs to the outputs of a PCD2.W4 D/A converter module.

One function box PCD2.W4 must be used for each PCD2.W4 card used.

The parameter Add is the base address of the card. E.g.: O 16. Output 240 must never be used for analogic module.

4.4.14.4 Analog Input/Output module PCD2.W5 (12 Bit)



[_anad2w5]

Inputs:	= i0	integer value
	= i1	integer value
Outputs:	= o0	integer value
	= o1	integer value

This function box outputs the A/D conversion of the input channels of the PCD2.W5 analogic card, and transfers its inputs to the D/A outputs of the same module. It has 2 inputs which are related to output o0 and o1 of the PCD2.W5, and it has 2 outputs which are associated to input i0 and i1 of the PCD.W5.

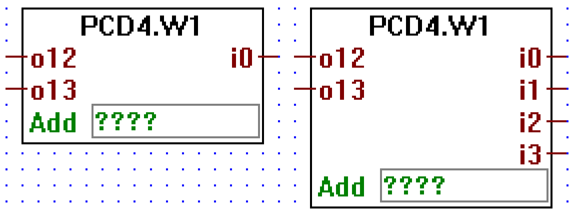
Each time this function box is executed one input channel is converted. When the conversion of an input channel is not ready its last valid conversion is output. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD2.W5 must be used for each PCD2.W5 card used.

The parameter Add is the base address of the card. E.g.: O 16. Output 240 must never be used for analogic module (conflict with the watchdog).

4.4.14.5 Analog Input/Output module PCD4.W1 (12 Bit)

[_anad4w1]



Inputs: = i0
¦ Integer values
= i3

Outputs = 012
¦ Integer values
= 013

This function box outputs the A/D conversion of the input channels of the PCD4.W1 analogic card, and transfers its inputs to the D/A outputs of the same module. It has 2 inputs o12 and o13 which are related to output 12 and 13 of the PCD4.W1, and it has up to 4 outputs labeled i0 to i3 which are associated to input 0 to 3 of the PCD4.W1.

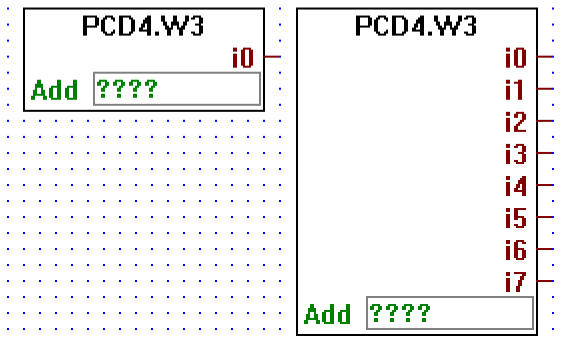
Each time this function box is executed one input channel is converted. When the conversion of an input channel is not ready its last valid conversion is output. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD4.W1 must be used for each PCD4.W1 card used.

The parameter Add is the base address of the card. E.g.: O 16. Output 240 and output 496 must never be used for analogic module.

4.4.14.6 Analog Input module PCD4.W3 (12 Bit + Sign)

[_anad4w3]



Inputs: = i0
¦ Integer values
= i7

This function box outputs the A/D conversion of the input channels of a PCID4.W3 analogic card. It has up to 8 outputs labeled i0 to i7 which are associated to input 0 to 7 of a PCD4.W3 analogic input card.

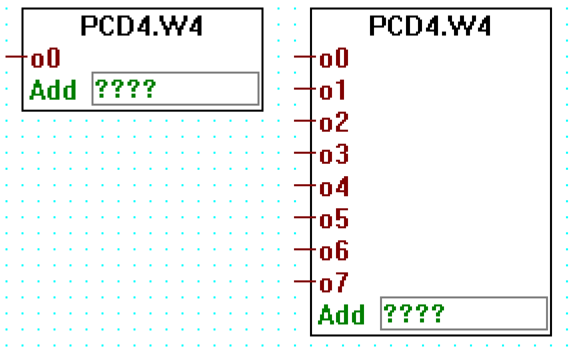
Each time this function box is executed one input channel is converted. When the conversion of an input channel is not ready its last valid conversion is output. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD4.W3 must be used for each PCD4.W3 card used.

The parameter Add is the base address of the card. E.g.: O 16. Output 240 and output 496 must never be used for analogic module.

4.4.14.7 Analog Output module PCD4.W4 (8 Bit)

[_anad4w4]



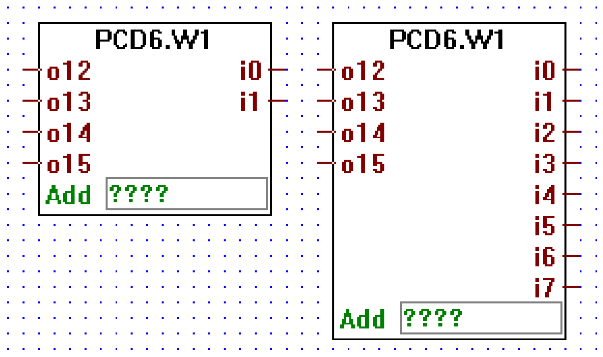
Outputs: = o0
¦ Integer values
= o7

This function box transfers its 1 to 8 inputs to the outputs of a PCD4.W4 D/A converter module.

One function box PCD4.W4 must be used for each PCD4.W4 card used. The parameter Add is the base address of the card. E.g.: O 16. Output 240 and Output 496 ust never be used for analogic module.

4.4.14.8 Analog Input / Output module PCD6.W1 (12 Bit)

[_anad6w1]



Outputs: = o12
¦ Integer values
= o15

Inputs: = i0
¦ Integer values
= i7

This function box outputs the A/D conversion of the input channels of a PCD6.W1 analogic card, and transfers its inputs to the D/A outputs of the same module. It has 4 inputs o12 to o15 which are related to output 12 to 15 of the PCD6.W1, and it has up to 8 outputs labeled i0 to i7 which are associated to input 0 to 7 of the PCD6.W1.

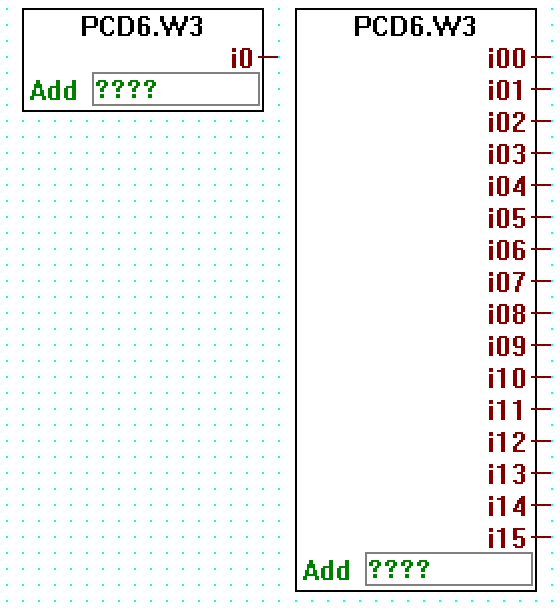
Each time this function box is executed one input channel is converted. When the conversion of an input channel is not ready its last valid conversion is output. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD6.W1 must be used for each PCD6.W1 card used.

The parameter Add is the base address of the card. E.g.: O 16. The last address available of each rack must never be used for analogic module card, i.e.: rack base address + 240.

4.4.14.9 Analog Input module PCD6.W3 (12 Bit + Siggn)

[_anad6w3]



Inputs: = i00
¦ Integer values
= i15

This function box outputs the A/D conversion of the input channels of a PCD6.W3 analogic card. It has up to 16 outputs labeled i0 to i15 which are associated to input 0 to 15 of a PCD6.W3 analogic input card.

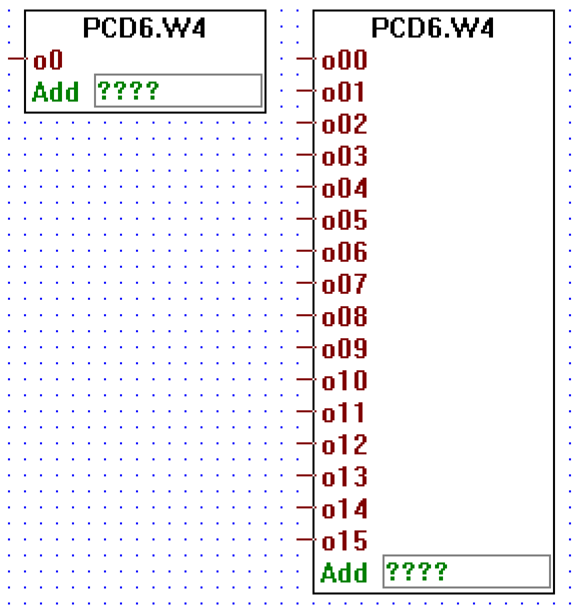
Each time this function box is executed one input channel is converted. When the conversion of an input channel is not ready its last valid conversion is output. If a channel has never been converted since the system initialisation, it will output 0.

One function box PCD6.W3 must be used for each PCD6.W3 card used.

The parameter Add is the base address of the card. E.g.: O 16. The last address available of each rack must never be used for analogic module card, i.e.: rack base address + 240.

4.4.14.10 Analog Output module PCD6.W4 (8 Bit)

[_anad6w4]



Outputs: = o00
¦ Integer values
= o15

This function box transfers its 1 to 16 inputs to the outputs of a PCD6.W4 D/A converter module.

One function box PCD6.W4 must be used for each PCD6.W4 card used.

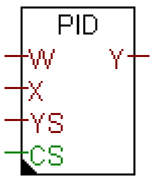
The parameter Add is the base address of the card. E.g.: O 16. The last address available of each rack must never be used for analogic module card, i.e.: rack base address + 240.

4.4.15 Regulation (PID control) [sfupregu]

4.4.15.1 PID FBox

PID FBox

[_regpid3]



Inputs:	W:	Set Point	integer format
	X:	Actual Value	integer format
	YS:	Cold Start Set Point	integer format
	CS:	Cold Start Signal	binary format
Output:	Y:	Manipulated Value	integer format

The following description is based on the assumption that the reader has a good knowledge of control engineering.The PID-instruction of the PCD is explained in the manual "PCD Series Reference Guide".

Cold start and manual mode:

YS ist the set point which is used from the system program as cold start value for Y. If the CS signal is dynamised then the regulation will continue after a cold start. If CS is high for a longer time, YS is transmitted to Y on each sampling signal. The regulation is out of use and may be continued in manual mode.

The following 6 parameters can be defined in the adjust window:

Sampling rate:
Defines the sampling rate at which the PID will take its sample. The units of time used in the variable is the same as the one which was defined for the timers. See also instruction DEFTB.

Proportional Factor P:
The factor P determine the proportional (amplification) characteristic of the regulator. When calculating, only the lower 16 bits are used (0..65535). The proportional factor is determined as follows:

Fp = (1/Xp)*256
with Xp: Proportional band.

Integral Factor Fi:
This factor determines the integral characteristic of the regulator. When calculating, only the 16 lower bits are used (0..65535). The Integral factor is determined as follow:

Fi = (T0/Ti) * 256
with T0: sampling time of the PID instruction
Ti: integral time.

Derivative Factor Fd:
This factor determines the derivative characteristic of the regulator. When calculating, only the 16 lower bits are used (0..65535). The Derivative factor is determined as follow:

Fd= (Td/T0) * 256
with T0: sampling time of the PID instruction
Td: derivative time.

Dead Range Dr:
The dead range defines the range in which the variations of the controlled variable may occur without causing a modification of the Manipulated variable MV.

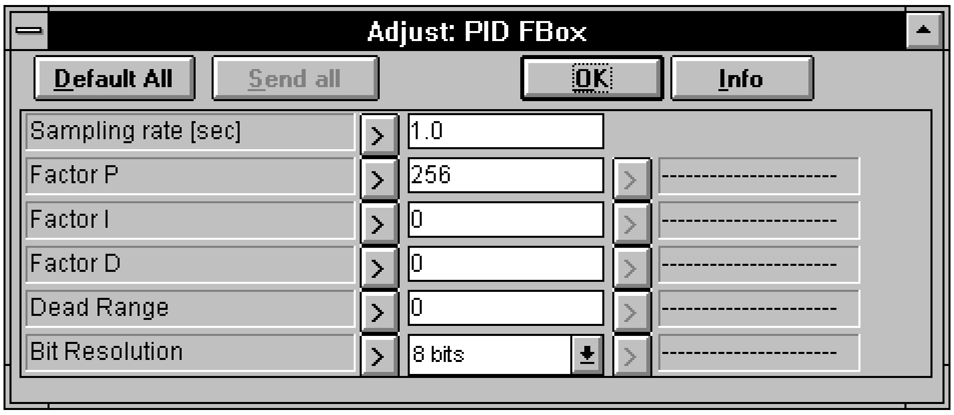
Cold Start Ys:
This value is used as starting value for Yn by the system program. As soon as the user program writes a value other than 0 to the cold start register, a cold start calculation is made and Yn is set with Ys, and the other internal values are Note: In order to achieve a smooth transition, Ys may be set equal to the currently used controlled variable.

Bit resolution:
The maximum of all manipulated variables are determined by the resolution.

8 bits: 0..255; 12 bits: 0..4095; 16 bits: 0..65,535

The resolution is mostly defined by the analog module used for the Result variable output.

Adjust window:



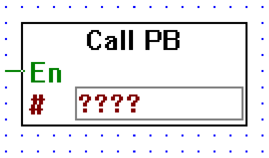
Sampling rate [sec]	1.0
Factor P	256
Factor I	0
Factor D	0
Dead Range	0
Bit Resolution	8 bits

4.4.16 User-defined functions [sfupuser]

A comfortable tool to program own functions is not yet available for this version.

To edit often used program parts it is recommended to program such parts as PB or FB in FUPLA or IL and call them conditionally or unconditionally from the FUPLA.

4.4.16.1 Call PB

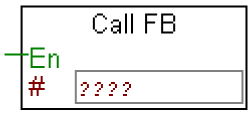


[_call_pb]

Inputs:	– En:	Enable	binary format
	#	PB address	integer format

Calls the PB number # (0 - 299) when the input En is HIGH.

4.4.16.2 Call FB

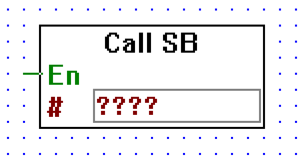


[_call_fb]

Inputs:	– En:	Enable	binary format
	#	FB address	integer format

Calls the FB number # (0 - 999) when the input En is HIGH.

4.4.16.3 Call SB



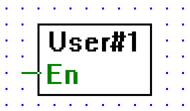
[_call_sb]

Inputs:	– En:	Enable	binary format
	#	SB address	integer format

Calls the SB number # (0 - 31) when the input En is HIGH.

4.4.16.4 User block 1

User block 1



[_user_1]

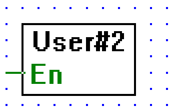
Input: – En: Enable binary format

Provides a way to include user code. When the binary input En (Enable) is actived the user code is executed. The user code must be put in the include file "user_1.h".

This solution does not provide a clean way of passing parameters from the FUPLA program to the user code, however future versions will. We strongly suggest not to use this scheme too intensively, future version will probably not support it.

4.4.16.5 User block 2

User block 2



[_user_2]

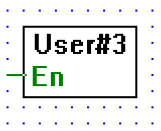
Input: – En: Enable binary format

Provides a way to include user code. When the binary input En (Enable) is actived the user code is executed. The user code must be put in the include file "user_2.h".

This solution does not provide a clean way of passing parameters from the FUPLA program to the user code, however future versions will. We strongly suggest not to use this scheme too intensively, future version will probably not support it.

4.4.16.6 User block 3

User block 3



[_user_3]

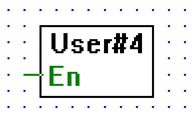
Input: – En: Enable binary format

Provides a way to include user code. When the binary input En (Enable) is actived the user code is executed. The user code must be put in the include file "user_3.h".

This solution does not provide a clean way of passing parameters from the FUPLA program to the user code, however future versions will. We strongly suggest not to use this scheme too intensively, future version will probably not support it.

4.4.16.7 User block 4

User block 4



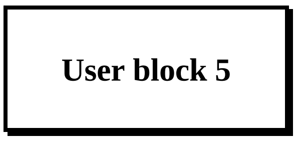
[_user_4]

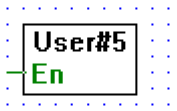
Input: – En: Enable binary format

Provides a way to include user code. When the binary input En (Enable) is actived the user code is executed. The user code must be put in the include file "user_4.h".

This solution does not provide a clean way of passing parameters from the FUPLA program to the user code, however future versions will. We strongly suggest not to use this scheme too intensively, future version will probably not support it.

4.4.16.8 User block 5





[_user_5]

Input: – En: Enable binary format

Provides a way to include user code.
When the binary input En (Enable) is actived the user code is executed.
The user code must be put in the include file "user_5.h".

This solution does not provide a clean way of passing parameters from the FUPLA program to the user code, however future versions will. We strongly suggest not to use this scheme too intensively, future version will probably not support it.

Notes :

4.4.17 Serial communications (mode "D") [sfupcomm]

4.4.17.1 Interface parameter assignment for a serial interface

Interface parameters

SASI

[_comsasi]

Initialization of serial communications: SASI

On PCD start-up, a serial interface is assigned with the defined communications mode. Each interface utilized in a PCD must be assigned with this function (a separate SASI function for each channel). After switching off and on again, or after a restart cold, the channels are assigned as soon as the PCD is switched into RUN, i.e. as soon as the restart cold routine XOB 16 has been executed.

Parameters:
Text number: Each SASI function uses a text (no. 0...3999). Care should be taken that the text has not already been used somewhere else in the program, or in any of the modules which are linked.

Channel: Serial interface number 0...3. For the PCD2, PCD4 and PCD6.M540, channel 0 is the programming unit interface (PGU). If it is assigned with SASI, this interface no longer functions for programming.

Mode:
Mode "None" allows assignment to be disabled, without losing the function parameters.

Mode "Text" is used to execute assignment in mode C. The texts and the user program in which these texts are used must be created with the PG3 Programming Utilities. Modes "Master" or "Slave" are not significant. Assignment always takes place in mode MC0.

Modes D, SBUS 0 and SBUS 1 provide important advantages when used with FUPLA, because of the "Communications" family's functions for transmitting and receiving boolean and integer values.

With mode D (point-to-point) one station must always be assigned as master (client) and the other as a slave (server). All transmit and receive functions must be programmed at the master station.

In S-Bus mode (multi-point connection) only one station can be assigned as master. All others are slave stations. If the master is a control system, it takes on the role of master. All other stations are slave stations here too. The transmit and receive functions cannot be used by the slave stations. For further information on communications modes, consult the PCD user manuals.

Baud rate:
This must be the same for all stations on the bus. The baud rate is to be selected in accordance with the quality of the bus and the potential outputs of the apparatus connected (PCD driver, repeater, modem, cable, etc.).

Number of bits, parity and stop-bit:
These settings must be the same for all stations connected. In SBUS mode these parameters are not used.

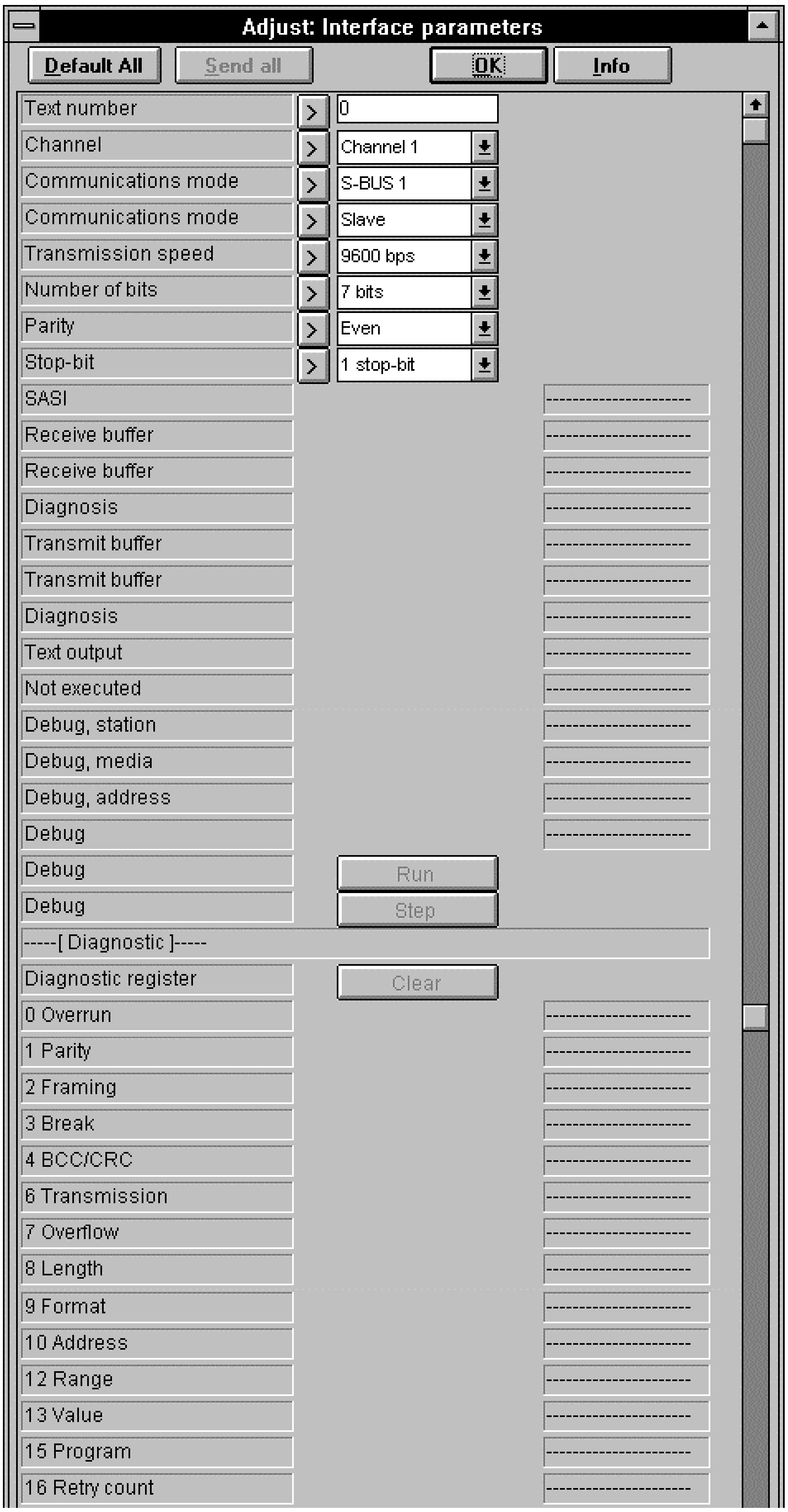
Diagnostic flag and diagnostic register:
See PCD user manuals.

The SASI online parameter ("OK" or "Error") displays whether the SASI function was executed correctly when the PCD was switched on. If there is an error, communication is not possible on this channel.

The following are potential sources of error:

- the firmware does not support the mode selected (especially SBUS)
- the CPU does not have access to this channel
- the CPU cannot process the (high) baud rate (mainly 38,400 bps)
- the channel has been assigned more than once (e.g. by a linked program)
- syntax error in the assignment text. This error should not arise with the present software package, unless the SASI text has been modified manually (e.g. using the debugger).

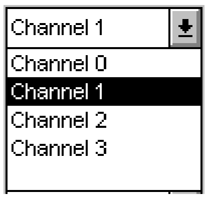
Debug:
The debug function allows the correct operation of communications to be monitored in the master station. If communications are running, there is a cyclical display of stations, media addressed in sequence, and diagnostic data. To obtain better control of diagnostic data and analyse a connection problem, communications can be interrupted with an initial click on the "Step" button. Each subsequent click triggers the next communications instruction (transmit or receive). With the "Run" button, communications can be made to continue normally. The information items "Station", "Media" and "Address" permit identification of the current transmit or receive function.

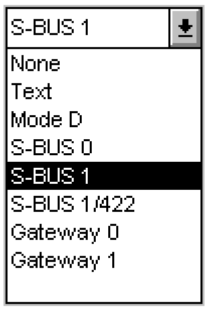


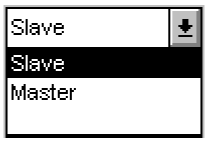
Adjust: Interface parameters

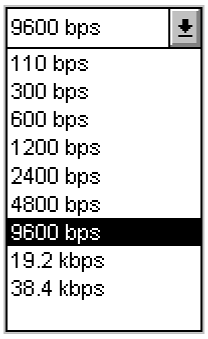
Default All | Send all | OK | Info

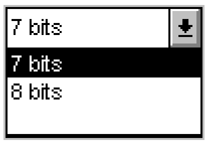
Text number	0
Channel	Channel 1
Communications mode	S-BUS 1
Communications mode	Slave
Transmission speed	9600 bps
Number of bits	7 bits
Parity	Even
Stop-bit	1 stop-bit
SASI	---------------------
Receive buffer	---------------------
Receive buffer	---------------------
Diagnosis	---------------------
Transmit buffer	---------------------
Transmit buffer	---------------------
Diagnosis	---------------------
Text output	---------------------
Not executed	---------------------
Debug, station	---------------------
Debug, media	---------------------
Debug, address	---------------------
Debug	---------------------
Debug	Run
Debug	Step
-----[Diagnostic]-----	
Diagnostic register	Clear
0 Overrun	---------------------
1 Parity	---------------------
2 Framing	---------------------
3 Break	---------------------
4 BCC/CRC	---------------------
6 Transmission	---------------------
7 Overflow	---------------------
8 Length	---------------------
9 Format	---------------------
10 Address	---------------------
12 Range	---------------------
13 Value	---------------------
15 Program	---------------------
16 Retry count	---------------------

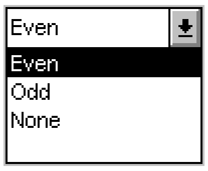


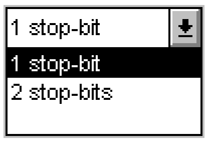




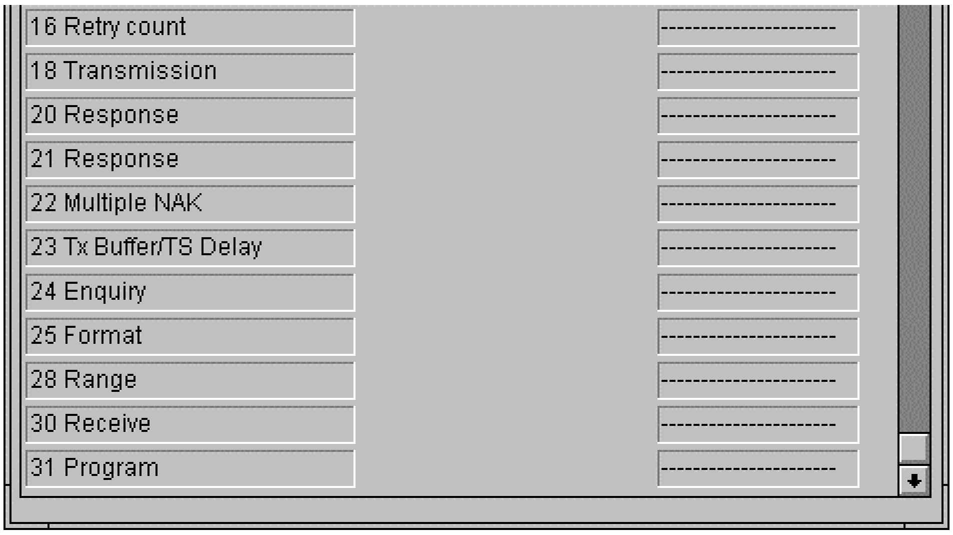








Continue of the adjust window on the next page.



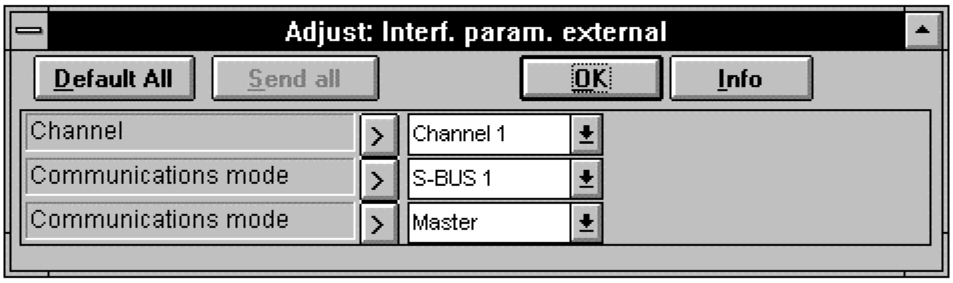
16 Retry count	---------------------
18 Transmission	---------------------
20 Response	---------------------
21 Response	---------------------
22 Multiple NAK	---------------------
23 Tx Buffer/TS Delay	---------------------
24 Enquiry	---------------------
25 Format	---------------------
28 Range	---------------------
30 Receive	---------------------
31 Program	---------------------

4.4.17.2 Interface parameters external

Interf. param. external

[_comsase]

If two or more FUPLA files in a project access the same serial interface, one (main) file must be used to execute "normal" assignment with the "Interface parameters" FBox. The other file(s) should then include the FBox shown here: "Interface parameters external", and set the same parameters as in the main file.

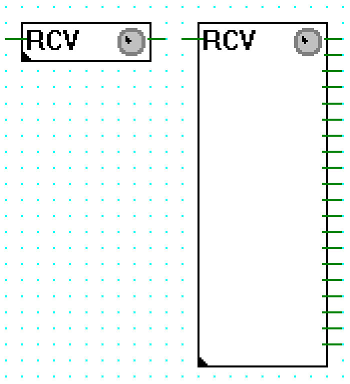


Channel	Channel 1
Communications mode	S-BUS 1
Communications mode	Master

4.4.17.3 Receive 1-20 Inputs/Outputs/Flags

Receive 1-20 I/O/F

[_comrxb]

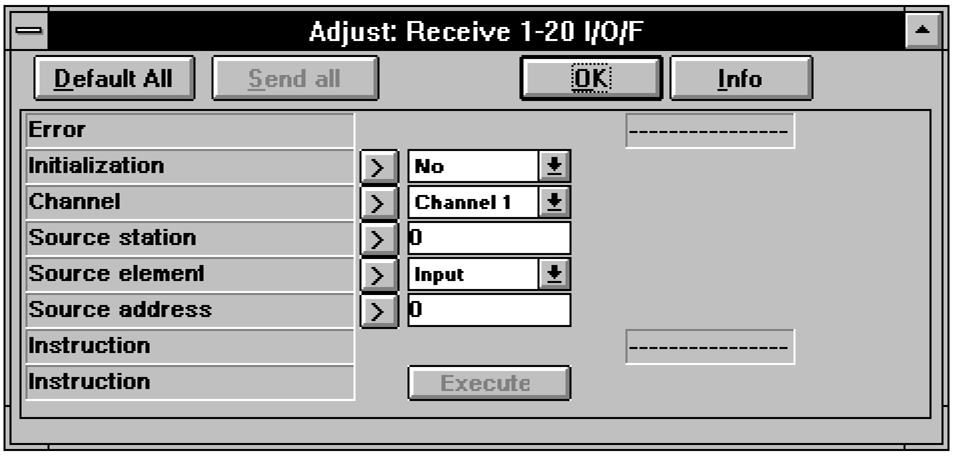


Input:	– Enable	binary format
Outputs:	– Bit values	binary format

If the Enable input = H (binary connection to the left of the function box), the data (boolean elements = inputs, outputs or flags) is transferred from the partner station to boolean elements in this station. The elements in this station are defined by the connections on the right-hand side of the function box.

The function can be extended from 1 up to 20 receive elements.

Additional settings, such as channel number, type of source element, etc., are made in the appropriate setting window.



Error	-----------------
Initialization	No
Channel	Channel 1
Source station	0
Source element	Input
Source address	0
Instruction	-----------------
Instruction	

Receive 1-20 I/O/F (detailed description)

This function can only be executed on a master PCD, which has been assigned with the SASI function in D or SBUS mode.

The function permits from 1 up to 20 x boolean elements (I/O/F) to be received from a slave station. Transmission takes place cyclically at the maximum possible speed for as long as the active signal is high (input "Send"). When the input receives a positive edge, at least one transmission takes place, even if the impulse is shorter than one communications transmission cycle.

Potential sources of error:

Assignment missing: The channel selected has been assigned wrongly or not at all (no SASI or invalid SASI). Assignment other than by the SASI function is not allowed.

Not master: The channel has not been assigned as master.

STXM: An error has been identified during execution of the STXM instruction. This ought not to arise when the present function is used, unless another routine also has access to this channel.

Diagnostics: A communications error has been identified. This can be analysed in detail using the diagnostic possibilities of the SASI function. The analysis can also take place in "Debug" mode.

Step: The channel has been put into "Step" operation by the "Debug" mode of the SASI function. Communication cannot take place until the channel is put into "Run" again.

Parameters:

Initialization: This option allows transmission during PCD start up, even when the binary active signal is low. This permits the elements of a slave station to be initialized after a halt of the
master station.

Channel: Channel number to use.

Station: Number of the slave station to which data are or have been transmitted. No significance in mode D.

Type and address: Base element address of the slave station where the elements transferred are to be stored. When more than one element is transferred, they are stored at subsequent addresses.

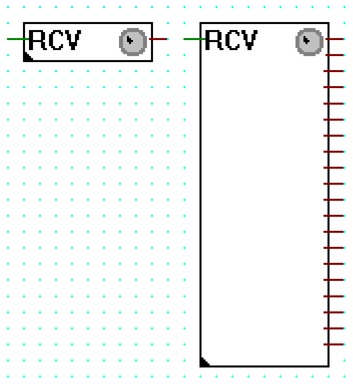
Two registers are always used for transmission of the hardware clock. Transmission must be arranged accordingly.

The "Instruction" button allows a transmission to be executed, even when the active signal is low.

4.4.17.4 Receive 1-20 Registers/Timers/Counters/Clock

Receive 1-20 R/T/C/Clock

[_comrxi]

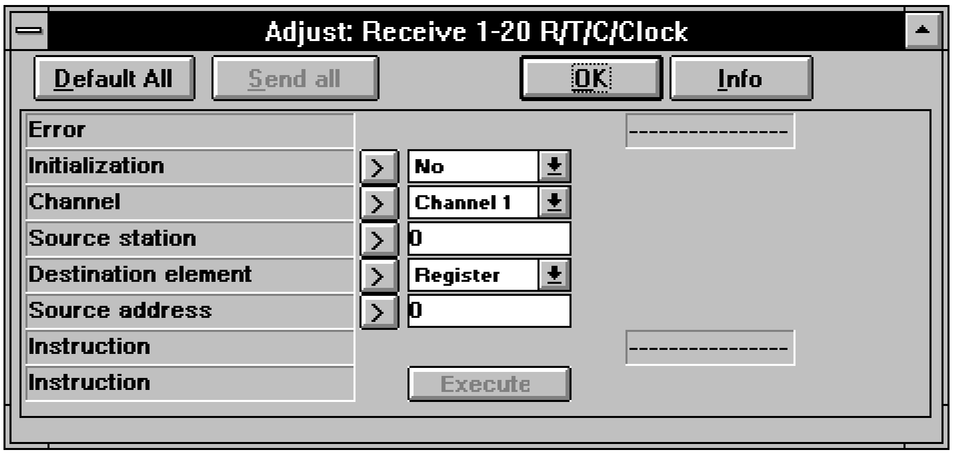


Input:	– Enable	binary format
Outputs:	= Values	integer format

If the Enable input = H (binary connection to the left of the function box), the values (numeric elements = registers, timers, counters or the hardware clock) are transferred from the partner station to the numeric elements in this station. The elements in this station are defined by the connections on the right-hand side of the function box.

The function can be extended from 1 up to 20 receive elements.

Additional settings, such as channel number, type of source element, etc., are made in the appropriate settings window.



Error		----------------
Initialization	No	
Channel	Channel 1	
Source station	0	
Destination element	Register	
Source address	0	
Instruction		----------------
Instruction		

Receive 1-20 R/T/C/Clock (detailed description)

This function can only be executed on a master PCD, which has been assigned with the SASI function in D or SBUS mode.

The function permits from 1 up to 20 x numeric elements (R/T/C/Clock) to be received from a slave station. Transmission takes place cyclically at the maximum possible speed for as long as the active signal is high (input "Send"). When the input receives a positive edge, at least one transmission takes place, even if the impulse is shorter than one communications transmission cycle.

Potential sources of error:

Assignment missing: The channel selected has been assigned wrongly or not at all (no SASI or invalid SASI). Assignment other than by the SASI function is not allowed.

Not master: The channel has not been assigned as master.

STXM: An error has been identified during execution of the STXM instruction. This ought not to arise when the present function is used, unless another routine also has access to this channel.

Diagnostics: A communications error has been identified. This can be analysed in detail using the diagnostic possibilities of the SASI function. The analysis can also take place in "Debug" mode.

Step: The channel has been put into "Step" operation by the "Debug" mode of the SASI function. Communication cannot take place until the channel is put into "Run" again.

Parameters:

Initialization: This option allows transmission during PCD start up, even when the binary active signal is low. This permits the elements of a slave station to be initialized after a halt of the
master station.

Channel: Channel number to use.

Station: Number of the slave station to which data are or have been transmitted. No significance in mode D.

Type and address: Base element address of the slave station where he elements transferred are to be stored. When more than one element is transferred, they are stored at subsequent addresses.

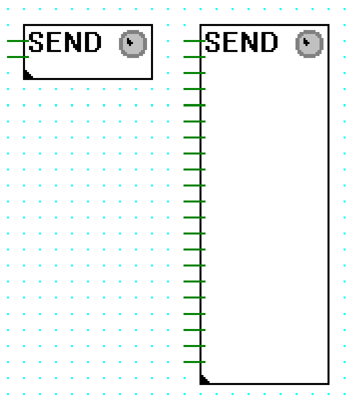
Two registers are always used for transmission of the hardware clock. Transmission must be arranged accordingly.

The "Instruction" button allows a transmission to be executed, even when the active signal is low.

4.4.17.5 Transmit 1-20 Inputs/Outputs/Flags

Transmit 1-20 I/O/F

[_comtxb]

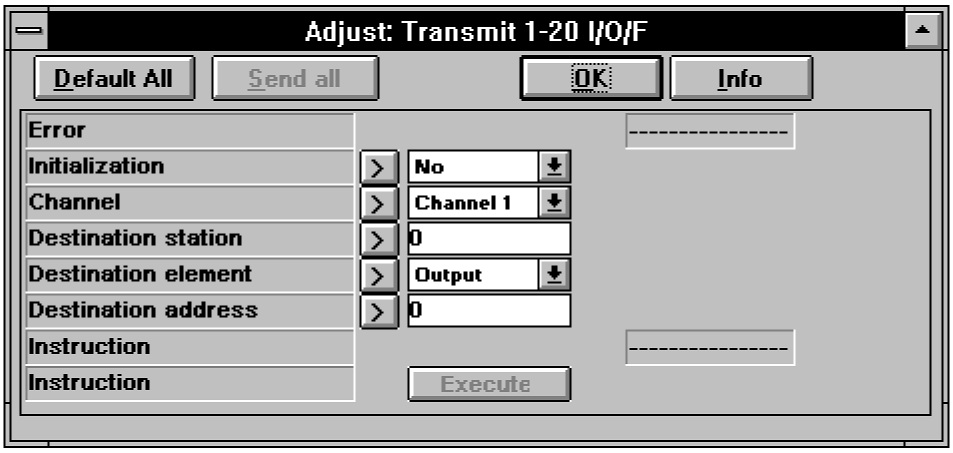


Inputs:	– Enable	binary format
	– Bit values	binary format

If the Enable input = H (binary connection at top left of the function box), the data (boolean elements = inputs, outputs or flags) is transferred from this station to a partner station. The elements in this station are defined by the connections on the left-hand side of the function box (2nd to 21st connections).

The function can be extended from 1 up to 20 transmit elements.

Additional settings, such as channel number, type of destination element, etc., are made in the appropriate settings window.



Error	----------------
Initialization	No
Channel	Channel 1
Destination station	0
Destination element	Output
Destination address	0
Instruction	----------------
Instruction	Execute

Transmit 1-20 I/O/F (detailed description)

This function can only be executed on a master PCD, which has been assigned with the SASI function in D or SBUS mode.

The function permits from 1 up to 20 x boolean elements (I/O/F) to be sent to a slave station. Transmission takes place cyclically at the maximum possible speed for as long as the active signal is high (input "Send"). When the input receives a positive edge, at least one transmission takes place, even if the impulse is shorter than one communications transmission cycle.

Potential sources of error:

Assignment missing: The channel selected has been assigned wrongly or not at all (no SASI or invalid SASI). Assignment other than by the SASI function is not allowed.

Not master: The channel has not been assigned as master.

STXM: An error has been identified during execution of the STXM instruction. This ought not to arise when the present function is used, unless another routine also has access to this channel.

Diagnostics: A communications error has been identified. This can be analysed in detail using the diagnostic possibilities of the SASI function. The analysis can also take place in "Debug" mode.

Step: The channel has been put into "Step" operation by the "Debug" mode of the SASI function. Communication cannot take place until the channel is put into "Run" again.

Parameters:

Initialization: This option allows transmission during PCD start up, even when the binary active signal is low. This permits the elements of a slave station to be initialized after a halt of the master station.

Channel: Channel number to use.

Station: Number of the slave station to which data are or have been transmitted. No significance in mode D.

Type and address: Base element address of the slave station where the elements transferred are to be stored. When more than one element is transferred, they are stored at subsequent addresses.

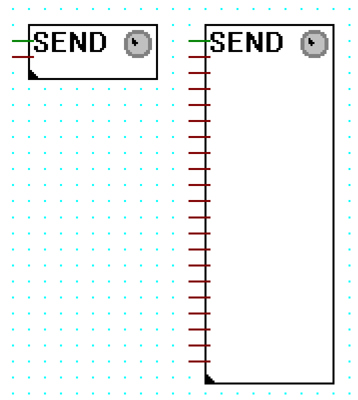
Two registers are always used for transmission of the hardware clock. Transmission must be arranged accordingly.

The "Instruction" button allows a transmission to be executed, even when the active signal is low.

4.4.17.6 Transmit 1-20 Registers/Timers/Counters/Clock

Transmit 1-20 R/T/C/Clock

[_comtxi]

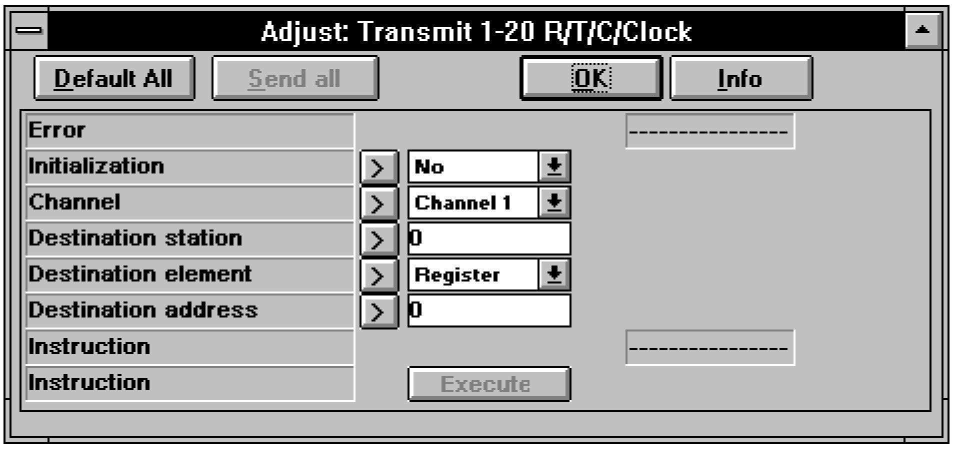


Inputs:	– Enable	binary format
	= Values	integer format

If the Enable input = H (binary connection at top left of the function box), the values (numeric elements = registers, timers, counters or the hardware clock) are transferred from this station to a partner station. The elements in this station are defined by the connections on the left-hand side of the function box (2nd to 21st connections).

The function can be extended from 1 up to 20 transmit elements.

Additional settings, such as channel number, type of destination element, etc., are made in the appropriate setting window.



Error	----------------
Initialization	No
Channel	Channel 1
Destination station	0
Destination element	Register
Destination address	0
Instruction	----------------
Instruction	

Transmit 1-20 R/T/C/Clock (detailed description)

This function can only be executed on a master PCD, which has been assigned with the SASI function in D or SBUS mode.

The function permits from 1 up to 20 x numeric elements (R/T/C/Clock) to be sent to a slave station. Transmission takes place cyclically at the maximum possible speed for as long as the active signal is high (input "Send"). When the input receives a positive edge, at least one transmission takes place, even if the impulse is shorter than one communications transmission cycle.

Potential sources of error:

Assignment missing: The channel selected has been assigned wrongly or not at all (no SASI or invalid SASI). Assignment other than by the SASI function is not allowed.

Not master: The channel has not been assigned as master.

STXM: An error has been identified during execution of the STXM instruction. This ought not to arise when the present function is used, unless another routine also has access to this channel.

Diagnostics: A communications error has been identified. This can be analysed in detail using the diagnostic possibilities of the SASI function. The analysis can also take place in "Debug" mode.

Step: The channel has been put into "Step" operation by the "Debug" mode of the SASI function. Communication cannot take place until the channel is put into "Run" again.

Parameters:

Initialization: This option allows transmission during PCD start up, even when the binary active signal is low. This permits the elements of a slave station to be initialized after a halt of the master station.

Channel: Channel number to use.

Station: Number of the slave station to which data are or have been transmitted. No significance in mode D.

Type and address: Base element address of the slave station where the elements transferred are to be stored. When more than one element is transferred, they are stored at subsequent addresses.

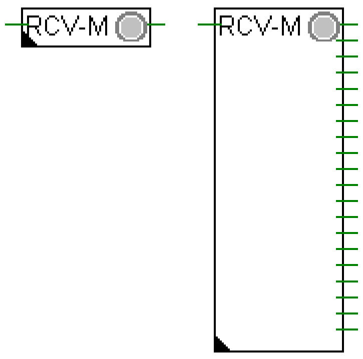
Two registers are always used for transmission of the hardware clock. Transmission must be arranged accordingly.

The "Instruction" button allows a transmission to be executed, even when the active signal is low.

4.4.17.7 Receive I/O/F from multiple stations

Receive I/O/F multiple

[_comrxbm]

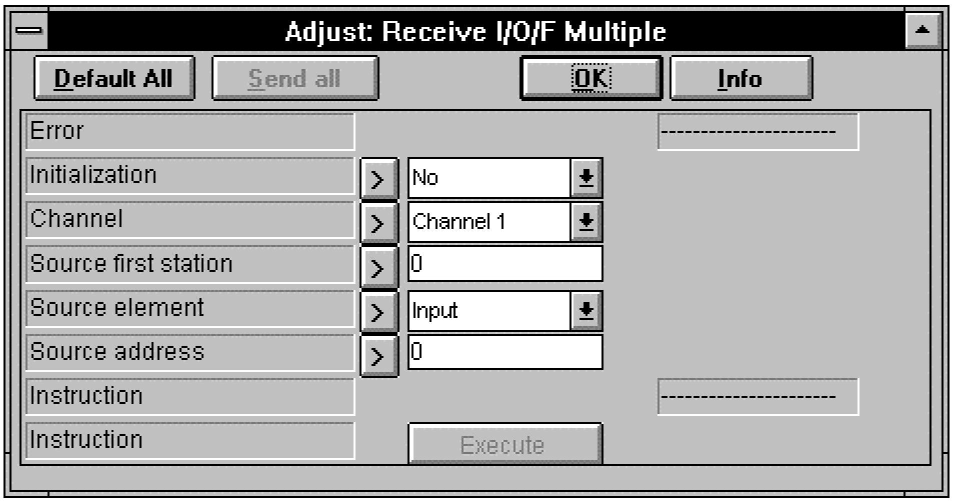


Input:	– Enable	binary format
Outputs:	– Values	binary format

If the Enable input = H (binary input of the Fbox), one value (binary elements = Input, Output or Flag) is transferred from each partner station in this station.

The function can be extended from 1 up to 20 output elements (one per station). The size of the Fbox defines the number of stations to be read. The first station is defined in the adjust window.

Additional settings, such as channel number, type of source element, etc., are made in the adjust window.



Error	---------------------
Initialization	No
Channel	Channel 1
Source first station	0
Source element	Input
Source address	0
Instruction	---------------------
Instruction	

This function can only be executed on a master PCD, which has been assigned with the SASI function in S-BUS Master mode.

The function permits to receive binary elements (I/O/F) from a number (1 up to 20) of successive slave stations. Transmission takes place cyclically at the maximum possible speed for as long as the active signal is high (binary input). When the input receives a positive edge, at least one transmission per station takes place, even if the impulse is shorter than one communications transmission cycle.

Potential sources of error:

Assignment missing: The channel selected has been assigned wrongly or not at all (no SASI or invalid SASI). Assignment other than by the SASI function is not allowed.

Not master: The specified channel has not been assigned as master.

Not S-BUS: The specified channel is not assigned in S-BUS. Only the S-BUS protocol can use this function.

SRXM: An error has been identified during execution of the SRXM instruction. This ought not to arise when the present function is used, unless another routine also has access to this channel.

Diagnostics: A communications error has been identified. This can be analysed in detail using the diagnostic possibilities of the SASI function.

To big: The Fbox has been stretched and so that the last station to be read (first station plus stretch index) is grater than 254. Only station number 0 to 254 are allowed for S-BUS Slaves.

Parameters:

Initialization: This option allows transmission during PCD start up, even when the binary active signal is low. This permits the elements of a slave station to be initialized after a halt of the master station.

Channel: Channel number to use.

Station: Number of the first slave station from which data are received.

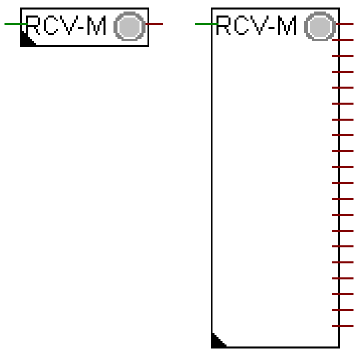
Type and address: Element address of the slave station where the elements are to be received. One element per station is transferred.

The "Instruction" button allows a transmission to be executed, even when the active signal is low.

4.4.17.8 Receive R/T/C from multiple stations

Receive R/T/C multiple

[_comrxim]

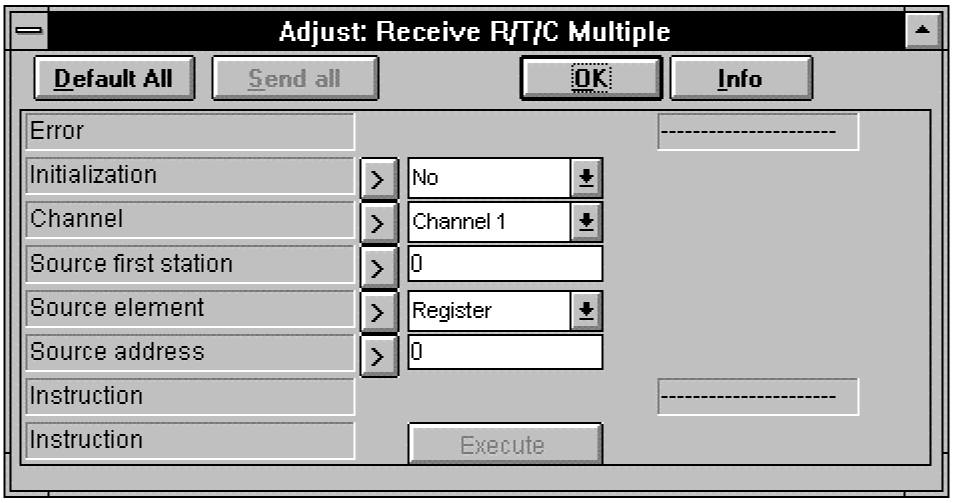


Input:	– Enable	binary format
Outputs:	– Values	integer format

If the Enable input = H (binary input of the Fbox), one value (numeric element = Register, Timers or Counters) is transferred from each partner station in this station.

The function can be extended from 1 up to 20 output elements (one per station). The size of the Fbox defines the number of stations to be read. The first station is defined in the adjust window.

Additional settings, such as channel number, type of source element, etc., are made in the adjust window.



Error	---------------------
Initialization	No
Channel	Channel 1
Source first station	0
Source element	Register
Source address	0
Instruction	---------------------
Instruction	

This function can only be executed on a master PCD, which has been assigned with the SASI function in S-BUS Master mode.

The function permits to receive numeric elements (R/T/C) from a number (1 up to 20) of successive slave stations. Transmission takes place cyclically at the maximum possible speed for as long as the active signal is high (binary input). When the input receives a positive edge, at least one transmission per station takes place, even if the impulse is shorter than one communications transmission cycle.

Potential sources of error:

Assignment missing: The channel selected has been assigned wrongly or not at all (no SASI or invalid SASI). Assignment other than by the SASI function is not allowed.

Not master: The specified channel has not been assigned as master.

Not S-BUS: The specified channel is not assigned in S-BUS. Only the S-BUS protocol can use this function.

STXM: An error has been identified during execution of the STXM instruction. This ought not to arise when the present function is used, unless another routine also has access to this channel.

Diagnostics: A communications error has been identified. This can be analysed in detail using the diagnostic possibilities of the SASI function.

To big: The Fbox has been stretched and so that the last station to be read (first station plus stretch index) is grater than 254. Only station number 0 to 254 are allowed for S-BUS Slaves.

Parameters:

Initialization: This option allows transmission during PCD start up, even when the binary active signal is low. This permits the elements of a slave station to be initialized after a halt of the master station.

Channel: Channel number to use.

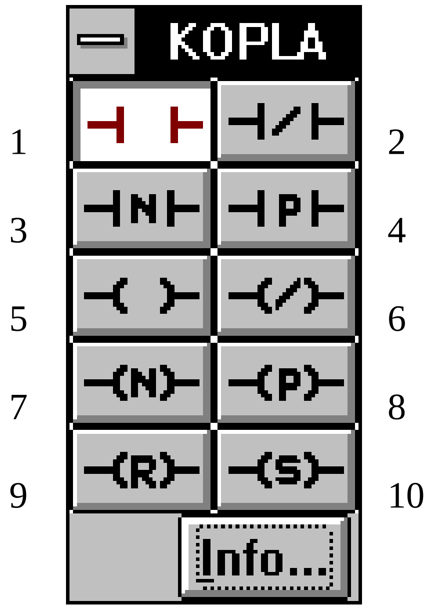
Station: Number of the first slave station from which data are received.

Type and address: Element address of the slave station where the elements are to be received. One element per station is transferred.

The "Instruction" button allows a transmission to be executed, even when the active signal is low.

4.5 The function families of the KOPLA (Ladder diagram)

The call of the KOPLA is done directly from the FUPLA toolbox and not via the "FBox Selection" menu.

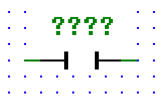


4.5.1	Contact	
4.5.2	Contact closed	
4.5.3	Contact negative	Negative edge
4.5.4	Contact positive	Positive edge
4.5.5	Coil	
4.5.6	Coil closed	
4.5.7	Coil negative	Negative edge
4.5.8	Coil positive	Positive edge
4.5.9	Coil reset	
4.5.10	Coil set	

KOPLA (Ladder diagram) (sfupkopl)

4.5.1 Contact

Contact

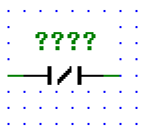


[_contact]

Normally Open Contact: The state of the left link is copied to the right link if the state of the associated Boolean variable is ON. Otherwise, the state of the right link is OFF.

4.5.2 Contact closed

Contact closed

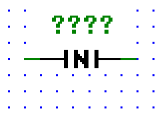


[_contactcl]

Normally Closed Contact: The state of the left link is copied to the right link if the state of the associated Boolean variable is OFF. Otherwise, the state of the right link is OFF.

4.5.3 Contact negative

Contact negative

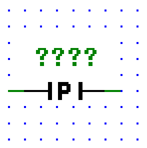


[_contactnv]

Negative Transition-Sensing Contact: The state of the right link goes ON when a transition of the associated variable from ON to OFF is sensed and when the state of the left link is ON. The state of the right link shall be OFF at all other time.

4.5.4 Contact positive

Contact positive

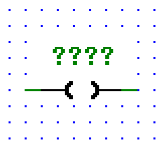


[_contactps]

Positive Transition-Sensing Contact: The state of the right link goes ON when a transition of the associated variable from OFF to ON is sensed and when the state of the left link is ON. The state of the right link shall be OFF at all other time.

4.5.5 Coil

Coil

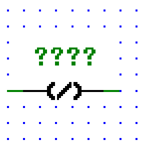


[_coil]

Coil: The state of the left link is copied to the associated Boolean variable and to the right link.

4.5.6 Coil closed

Coil closed

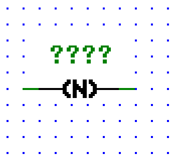


[_coilcl]

Negated Coil: The state of the left link is copied to the right link. The inverse of the state of the left link is copied to the associated Boolean variable.
i.e. if the state of the left link is OFF, then the state of the associated variable is set ON, and vice versa.

4.5.7 Coil negative

Coil negative

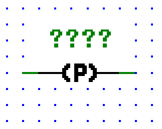


[_coilnv]

Negative Transition-Sensing Element: The state of the associated Boolean variable is ON from one evaluation of this element to the next when a transition of the left link from ON to OFF is sensed. The state of the left link is always copied to the right link.

4.5.8 Coil positive

Coil positive

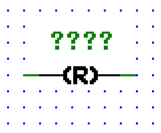


[_coilps]

Positive Transition-Sensing Element: The state of the associated Boolean variable is ON from one evaluation of this element to the next when a transition of the left link from OFF to ON is sensed. The state of the left link is always copied to the right link.

4.5.9 Coil reset

Coil reset

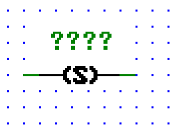


[_coilreset]

RESET (Unlatch) Coil: The associated Boolean variable is set to the OFF state when the left link is in the ON state, and remains reset until set by a SET coil.

4.5.10 Coil set

Coil set



[_coilset]

SET (Latch) Coil: The associated Boolean variable is set to the ON state when the left link is in the ON state, and remains set until reset by a RESET coil.

From :	Send back to :
Company : Department : Name : Address :	SAIA-Burgess Electronics Ltd. Bahnhofstrasse 18 CH-3280 Murten (Switzerland) http://www.saia-burgess.com
Tel. :	BA : Electronic Controllers
Date :	The FUPLA and the KOPLA function families PG4 - Version 1.3

If you have any suggestions concerning the SAIA® PCD, or have found any errors in this manual, brief details would be appreciated.

Your suggestions :