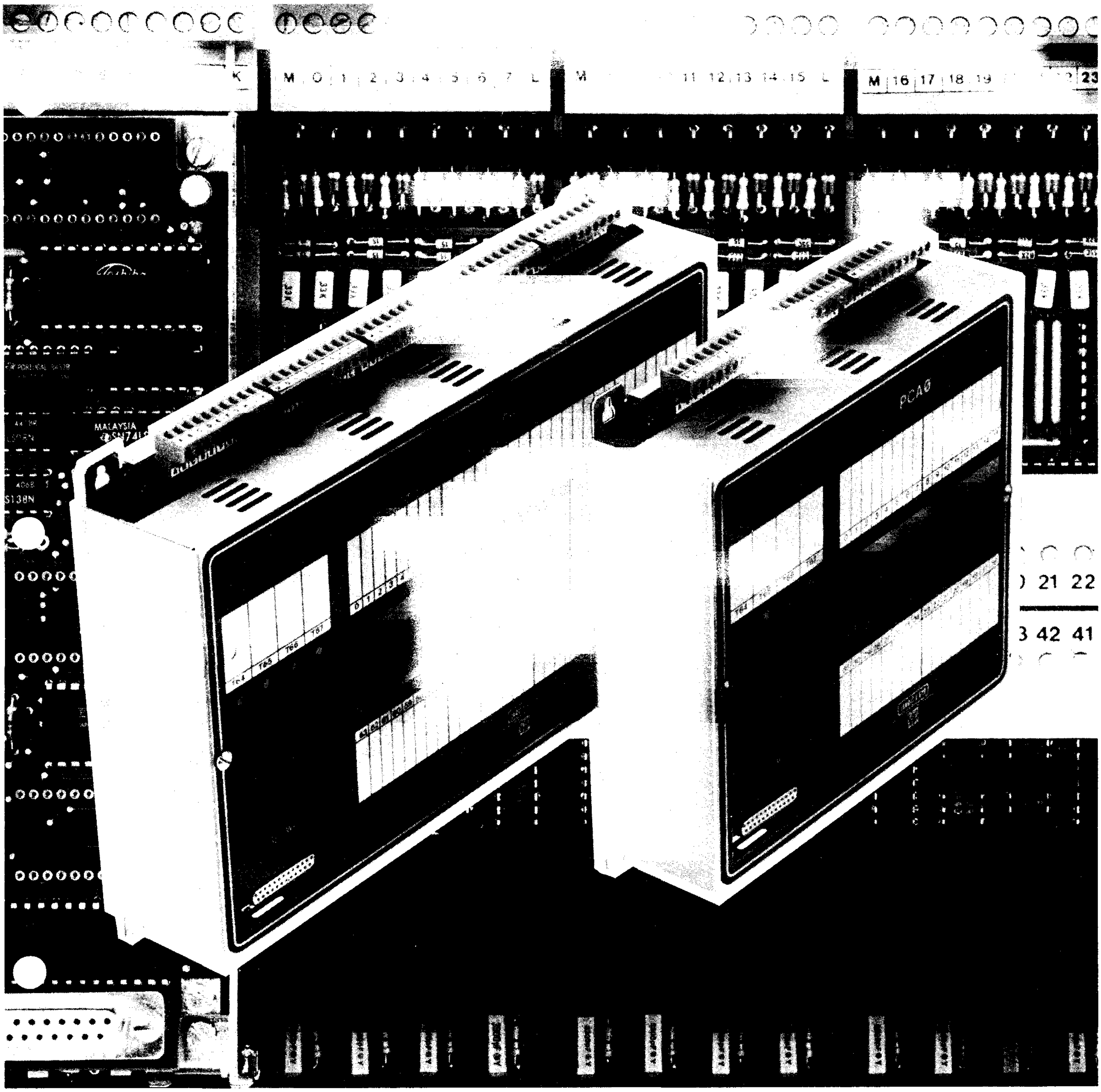
saia® pc
programmable controllers

LANDIS & GYR

SAIA

Manual PCA0-standard

Compact-PLC with high intelligence



English edition 26/79 E2

Series PCA0

The compact

highly-intelligent

programmable controller

- Four standard versions with up to 64 I + 0
- 4K user steps
- Programmable according to ladder diagram, logic diagram, flow-chart or functional diagram as per DIN-standard.
- Arithmetic instructions and security commands for permanent monitoring of the operation (watchdog and check sum).
- Easy beginning for learners but with big power reserve for the demanding PC-user.

01.03.1987

Selling price: sFr. 20.--

Who uses the PCAØ-manual?

We do not know how familiar you are with programmable controllers. Maybe you are a beginner or already an experienced PC-specialist.

This manual serves as a course, in order to give the beginner an easy introduction to the world of programmable controllers. Read especially the chapters 1 to 5 carefully before you start to work and do not let yourself be confused with the diagrams in the chapters 6 and 7. These are not important at the beginning if you have equipped yourself with the simulation and power supply unit PCA2.SØ5 (see chapter 5e).

In chapter 9 you will be gradually led up the staircase from A via B to C. For this, we use easy and clear examples, which can be collectively tested on your desk with the above-mentioned simulation unit.

If you do not know how to go on, please make use of the experience of our specialist in your vicinity or register for the next workshop.

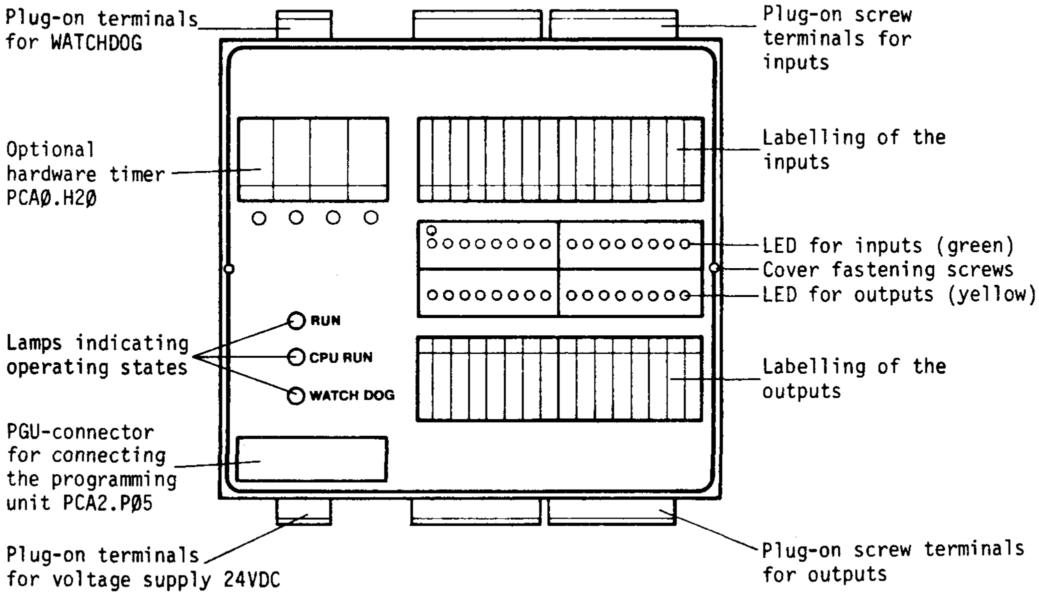
We wish you a lot of fun with the versatile PCAØ!

If you are a PC-specialist, you do not have to read all the information in chapter 9. In this case, concentrate on the instruction lists at the beginning of parts A, B and C. If you want to know more about these instructions, refer to the elaborated "Basic Manual".

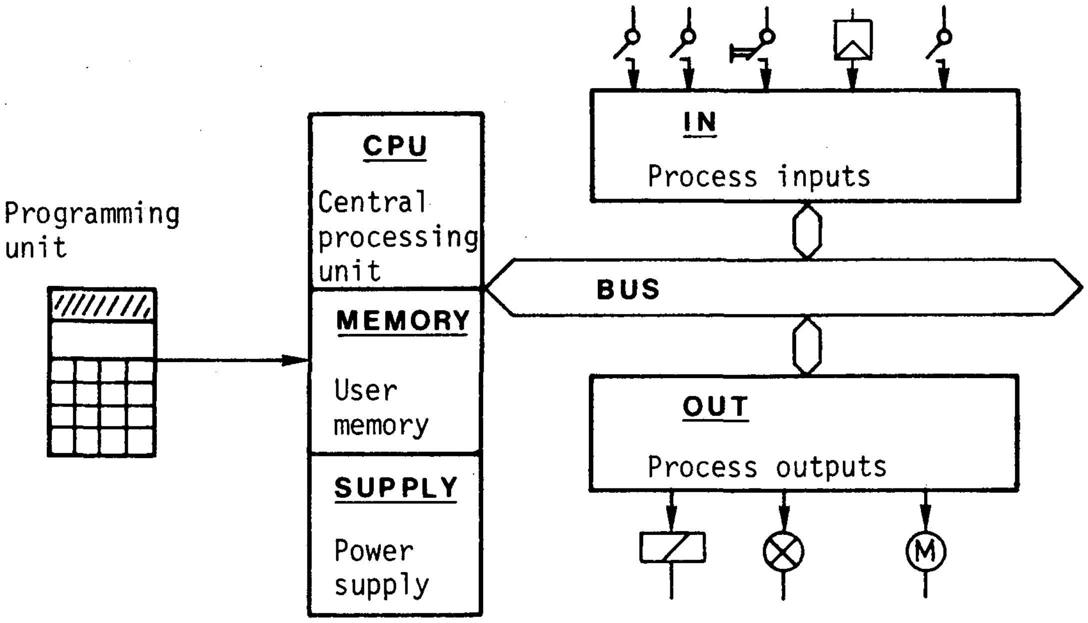
1. A look at the exterior and interior of the PCAØ

The PCAØ is the compact series of the SAIA-PC system family. The input and output assemblies are combined on a single pc-board. Owing to the high intelligence very simple as well as quite complex problems can be solved with the PCAØ.

On the exterior, the following functional parts are distinguished:



The following functional units can be seen inside:



The programming unit serves to transfer the user program to the user memory (RAM). The CPU executes this program, interrogates the states of the process inputs and controls the process outputs accordingly.

saia®pc

2. Common technical data

Microprocessor system	8Ø85-2
Cycle time per user instruction (average)	7Ø µs
Instruction set level 1H	32 basic instructions + 2Ø additional instructions incl. arithmetic
Number of parallel programs	16
Number of index registers	16 (1 per parallel program)
Number of subroutine levels	3
User memory	4K program lines (≘ 8K Bytes)
Volatile/non-volatile flags	477* + 235 = 712
Number of software counters and timers	64 addresses (C = 64, T = 32)
Counting capacity	65'535
Time ranges (time base Ø.1 or Ø.Ø1s)	Ø.1 (Ø.Ø1) to 65ØØ (65Ø)s
Hardware timer PCAØ.H2Ø	4 time ranges Ø.9/3.7/3Ø/24Øs
Connection of peripherals	via 25-pole PGU-connector
Operating modes	RUN, BREAK, STEP, MAN, PROG
Indicating lamps	LED for RUN/CPU RUN/WATCHDOG LED for I/O
Inputs (B9Ø)	galvanically connected, source or sink operation nominal +24VDC H = +19...+32V L = Ø...+ 4V I = 1ØmA, 24VDC, t_I = 9ms
Relay outputs (A21)	galvanically isolated, normally open contacts contact rating 3A, 25ØVAC AC1 3A, 24VDC DC1
Transistor outputs (B9Ø)	galvanically connected, positive switching Ø.5mA...Ø.5A, 5...36VDC
Supply voltage	24VDC ±2Ø%
Ambient temperature	Ø...+5Ø°C
High noise immunity	as per IEC 255-4/E5 class III, i.e. 25ØØV and IEC 8Ø1-4 class III (2ØØØV)

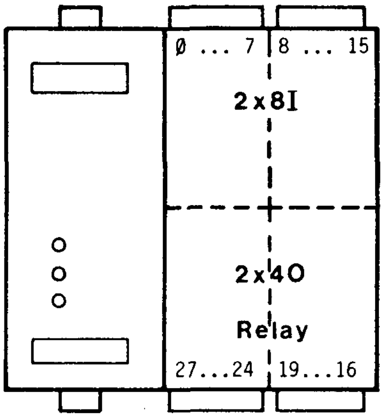
*) By inserting the jumper "NV", all flags and registers for timers and counters are made non-volatile.

3. The four standard versions

3.1 With relay outputs

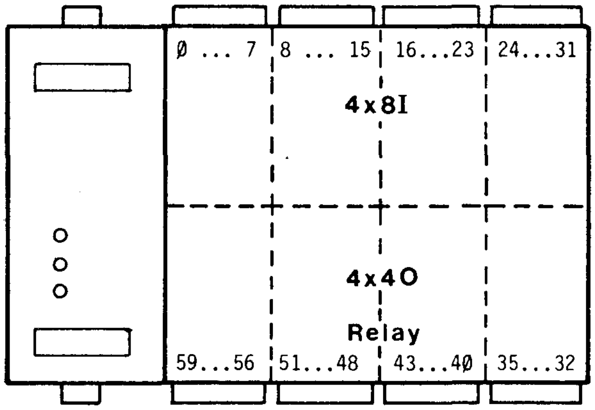
Type PCAØ.M12R M4

16 I, 24VDC
8 O, relay contact
max. 3A, 25ØVAC



Type PCAØ.M14R M4

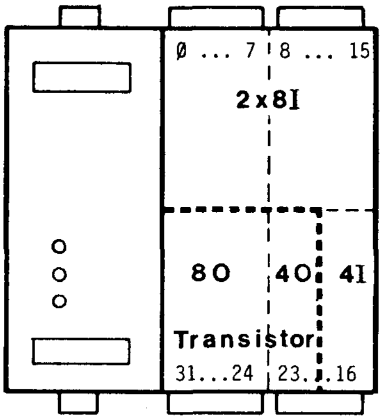
32 I, 24VDC
16 O, relay contact
max. 3A, 25ØVAC



3.2 With transistor outputs

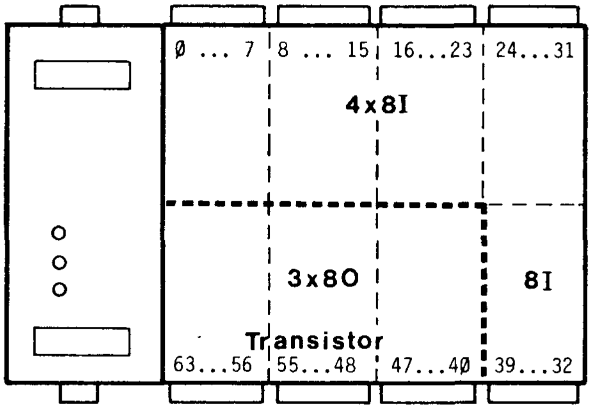
Type PCAØ.M12T M4

2Ø I, 24VDC
12 O, Ø.5A/24VDC



Type PCAØ.M14T M4

4Ø I, 24VDC
25 O, Ø.5A/24VDC

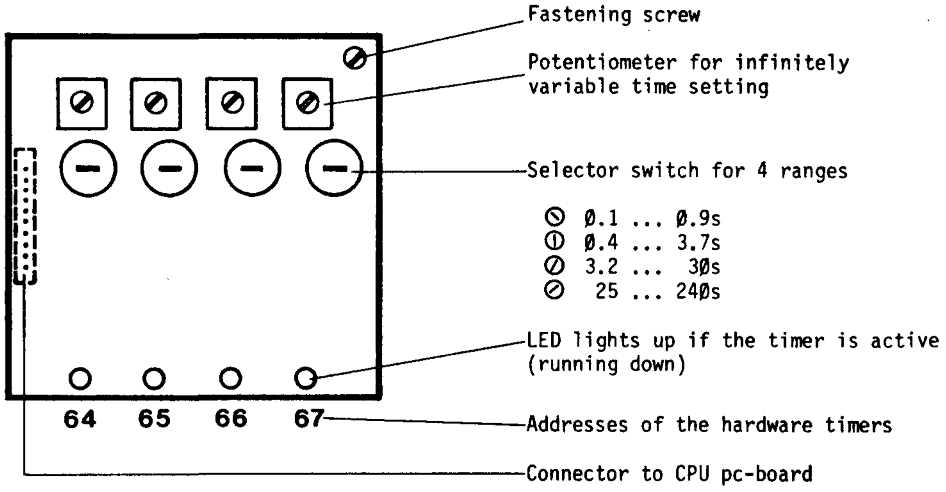


Detailed information on the inputs/outputs see chapter 7.

Please note: If a certain number of units are ordered, we will supply you, too, with a custom-made version. Please contact our nearest selling agency.

4. Important accessories

4.1 The hardware timer module PCAØ.H2Ø



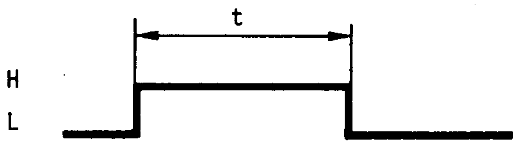
This module is an option and must be ordered separately. It allows easy setting of four time ranges in the RUN-mode independently of the program (e.g. for setting delay times).

The repetition accuracy is:

- under constant conditions Ø.1% of the time range set
- under extreme conditions 1% of the time range set
  (T = Ø...5Ø°C, U = 24V ±2Ø%)

For waiting for a time to elapse, set at timer 64, use the following simple program:

```
REO 64
SEO 64
WIH 64
```



The corresponding LED lights up while the timer is running down.

Please do not forget that this module is only necessary, if the 32 software timers included in every standard PCAØ do not suffice. The software timers can be modified in the range Ø.Ø1s to 65ØØs by the program or via 8 inputs by means of the BCD-switches in the RUN-mode (see example A8).

4.2 The user memory

Three different types of user memories with 4K (4Ø96) user steps each are available:

- RAM-chip 8464 on socket, order no. 4´5Ø2´4718´Ø

  This type of memory allows one to write, erase or overwrite a program as desired with the hand-held programming unit PØ5. In case of a voltage failure, the memory contents are stored in the CPU for approximately 2 months thanks to the buffer battery. The program, however, cannot be transported, as it is lost when the RAM-chip is removed.

- Buffered RAM-memory module type PCA1.R95

  Contrary to the RAM-chip, the program in this memory can be transported, as it is protected by an integrated electronic system and stored by a lithium battery for approximately 8 years.

  Programming with the hand-held programming unit PØ5.

- EPROM-chip 2764 on socket, order no. 4´5Ø2´4719´Ø

  In an EPROM a program is reliably stored for more than a decade. However, the program cannot be entered directly into the EPROM with the PCAØ. For this, the following possibilities are offered (ask for the special documentation):

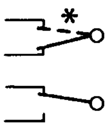
  a) with the EPROM-load module PCA2.P16
  b) with the universal programming unit PCA2.P21
  c) with the CPUs of the series PCA2 (M31 and M32)

  Every EPROM can be erased with an appropriate UV-light source almost as often as desired.

Depending on the user memory in use, the selection jumpers on the CPU must be inserted (see also figure in chapter 5).

for R95

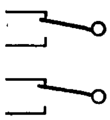
2764
8464/R95
2764/$\overline{\text{R95}}$
8464



Standard
factory setting

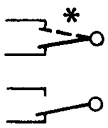
for EPROM 2764

2764
2764



for RAM 8464

8464
8464



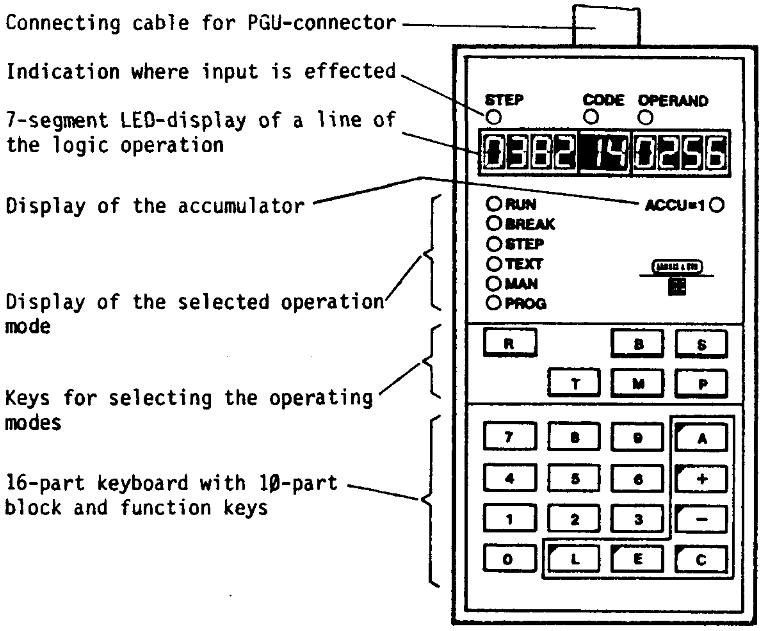
---

*) Position for write-protection

Please note: The jumpers should be repositioned only with the PC switched off.

4.3 The programming units

- The hand-held programming unit PCA2.PØ5

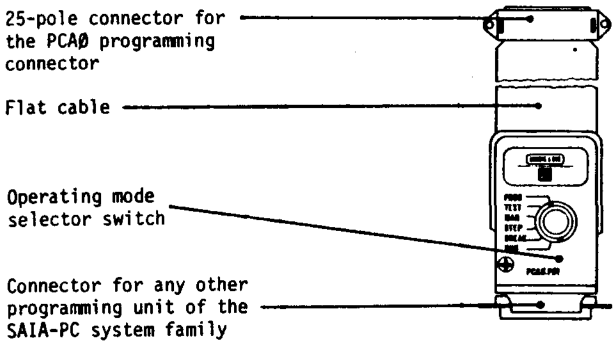


This handy programming unit was developed in particular for the series PCAØ, but it can also be used for the series PCA1 and PCA2.

All operating modes can be selected with keys. Programming is performed in the PROG-operating mode by means of a 1Ø-part keyboard in the easily understandable numerical code. All elements (inputs, outputs, timers, counters) can be interrogated or set in the "MAN"-operating mode.

All timer and counter values can be indicated in the RUN-mode. In the operating mode "STEP" a jump can be effected to any user step of the 4K-user memory. Finally, "BREAK" permits the program execution up to a set break-point and continuation in step-by-step operation. For details refer to "Operating modes" in chapter 8.

- The programming interface PCAØ.PØ1

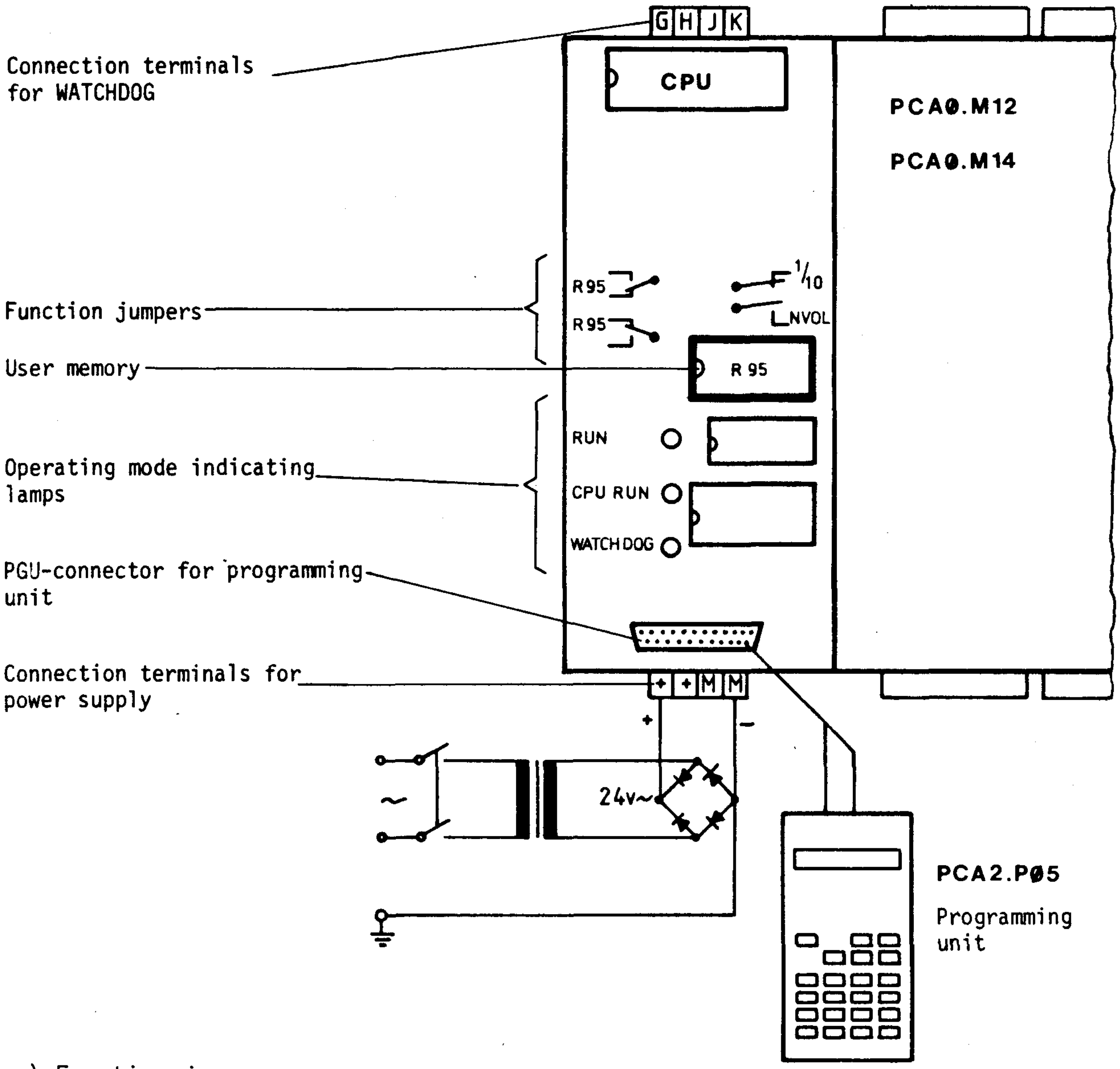


This interface also allows connection of all SAIA-PC programming units with the series PCAØ, namely the following:

- P1Ø hand-held programming unit with numerical code
- P18 hand-held computer with numerous possibilities
- P21 universal programming unit
- IBM-PC with SAIA-macro-assembler.

As a result, all upwards-compatible members of the SAIA-PC system family are available also for the PCAØ.

5. Brief instructions for operating the PCAØ



a) Function jumpers

① When delivered, the function jumpers are inserted as follows:

- Time base "1/1Ø" is inserted (for a time base of 1/1Øs)
- Flags and registers are non-volatile, when "NVOL" (non-volatile) is not inserted
- Jumpers for user memory as evident from the above figure apply to the buffered RAM-module PCA1.R95.

If the jumpers are not in these positions, they can be changed with a small screw-driver. In order to provide access to the CPU the cover needs to be removed by two screws.

The jumpers should be repositioned only with the PC switched off.

b) Power supply

② Take a transformer (for "playing" 2ØVA is enough) with a secondary voltage of 24VAC and connect the terminals + and M of the PCAØ via a bridge rectifier. (The PCA2.SØ5 simulation unit already contains this power supply, see section e).

③ A switch gives the advantage that by switching off the PCAØ all resettable elements and the STEP counter can be easily reset to their initial defined positions.

c) Installation of the user memory R95 and programming unit PØ5

④ The buffered RAM-module PCA1.R95 needs to be plugged onto the empty user socket in the specified position (notch on the left).

⑤ The programming unit PCA2.PØ5 is connected via the 25-pole PGU-connector.
If any other programming unit than PØ5 is used, the interface PCAØ.PØ1 needs to be interposed.

d) Program example "Blinker"

⑥ Switch on voltage supply. Yellow lamp "CPU RUN" blinks every 2s (1s on, 1s off).

⑦ Select the operating mode "PROG" by pressing the $\boxed{P}$-key of the programming unit (for at least Ø.5s). As a result, the red LED "PROG" on the PØ5 lights up.

⑧ Enter the following blinker program:

	STEP	CODE	OPERAND	Program in mnemonic code	
A,Ø,E	(ØØØØ) *	(ØØ)	(ØØØØ)		
E	(ØØØ1)	Ø2	256	STL	256
E	(ØØØ2)	14	256	STR	256
E	(ØØØ3)	ØØ	5		Ø.5s
E	(ØØØ4)	13	24/4Ø **	COO	24/4Ø **
E	(ØØØ5)	2Ø	1	JMP	1
E	(ØØØ6)	(ØØ)	(ØØØØ)		

*) The values in brackets do not have to be entered, but they are indicated.

**) For the small PCAØ.M12.. enter output 24, for the big PCAØ.M14.. enter output 4Ø.

⑨ Set program counter to zero:

Key sequence [S] operating mode STEP, as confirmation the red LED "STEP" lights up

[A] address, [Ø] [+]

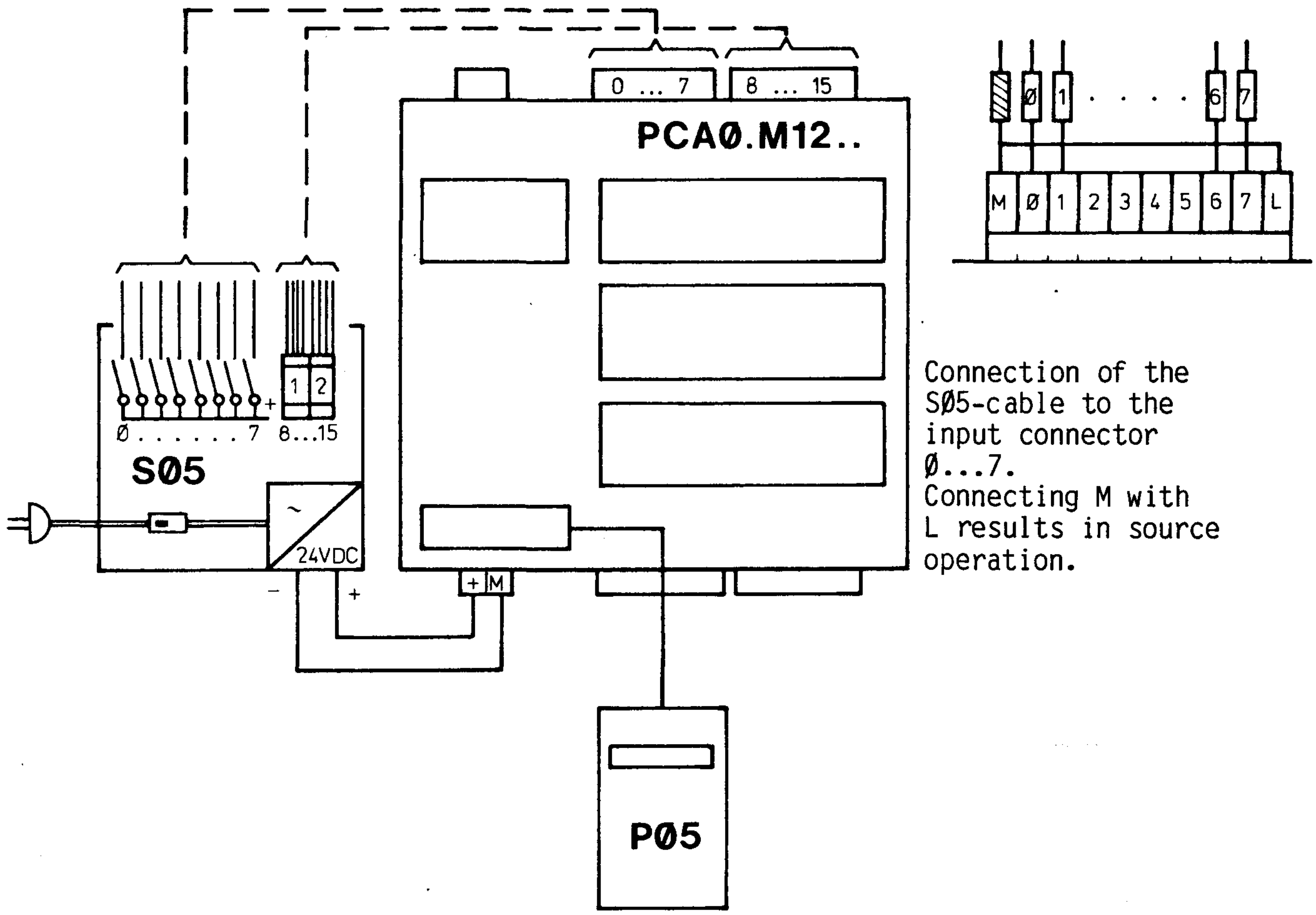
⑩ Select operating mode "RUN":

Press key [R] (RUN) for Ø.5s

→ Red LED "RUN" lights up on PØ5
→ Green lamp "RUN" lights up on PCAØ
→ Output 24 or 4Ø blinks Ø.5s on and Ø.5s off (frequency 1Hz)

e) Connection of the input simulation unit PCA2.SØ5

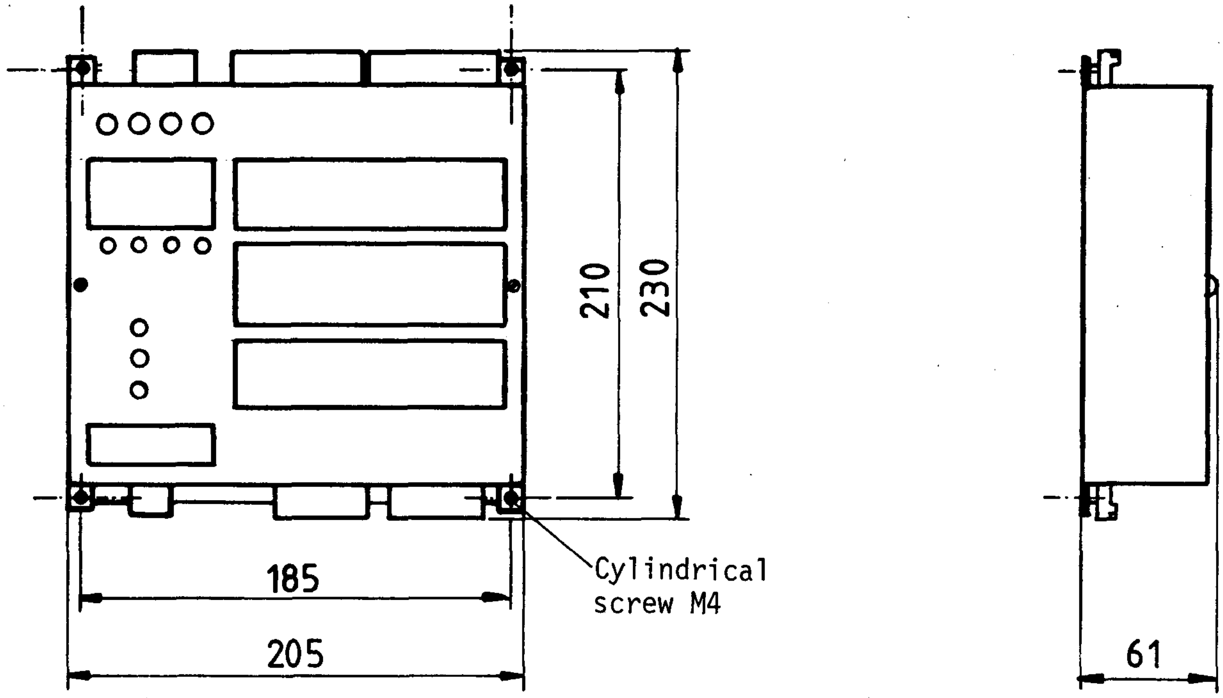
Including the I-simulation unit PCA2.SØ5 gives a complete set of programming and practicing devices which can be used to try out all examples of programs contained in this manual.

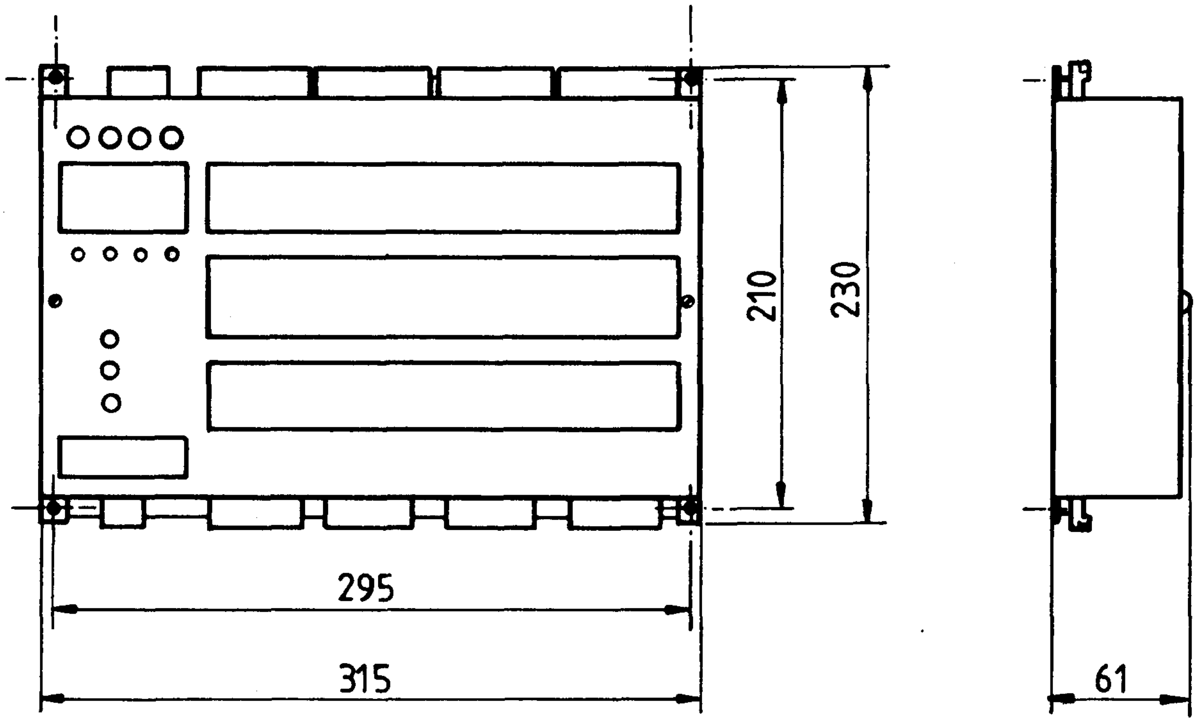


Connection of the SØ5-cable to the input connector Ø...7. Connecting M with L results in source operation.

Instead of the I-simulation unit PCA2.SØ5 the bigger PCA2.S1Ø can also be used with the connecting cable PCA1.K8Ø.

Dimension diagram of the PCAØ



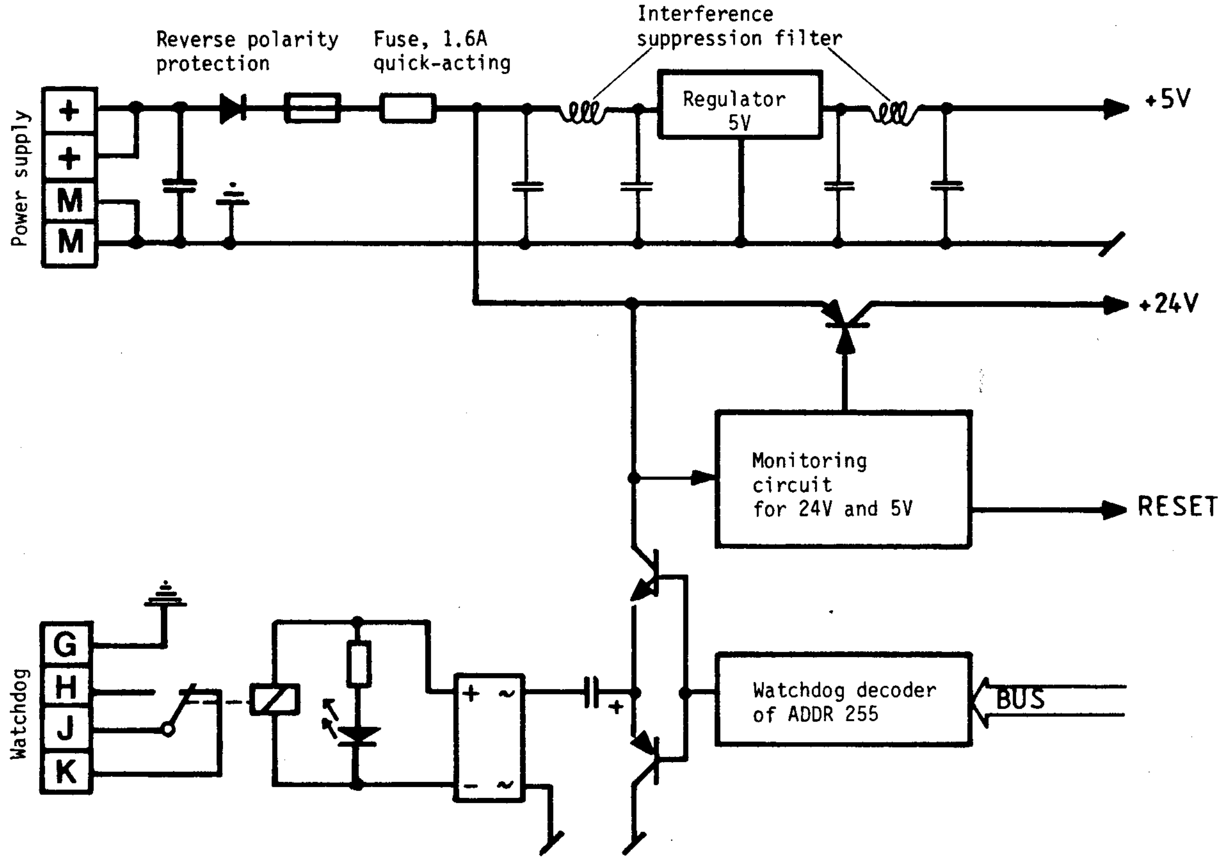


6. Detailed information on power supply and watchdog

6.1 Power supply of the PCAØ

Supply voltage U_{in}	24VDC smoothed or pulsating
Tolerance for U_{in}	
- general	±2Ø%
- for version with relays	pulsating voltage ±2Ø% (T_{amb} = Ø...5Ø°C)
	smoothed voltage +2Ø% / -5% (T_{amb} = Ø...35°C)
Supply current	
- PCAØ.M12T	max. Ø.5A (with PØ5 connected)
- PCAØ.M14R	max. Ø.9A (with PØ5 connected)
Fuse	1.6A quick-acting

Several components protect the PCAØ against interference voltages, wrong polarity and voltage drops. The 5V for supplying the electronic components is generated by means of a switching regulator.



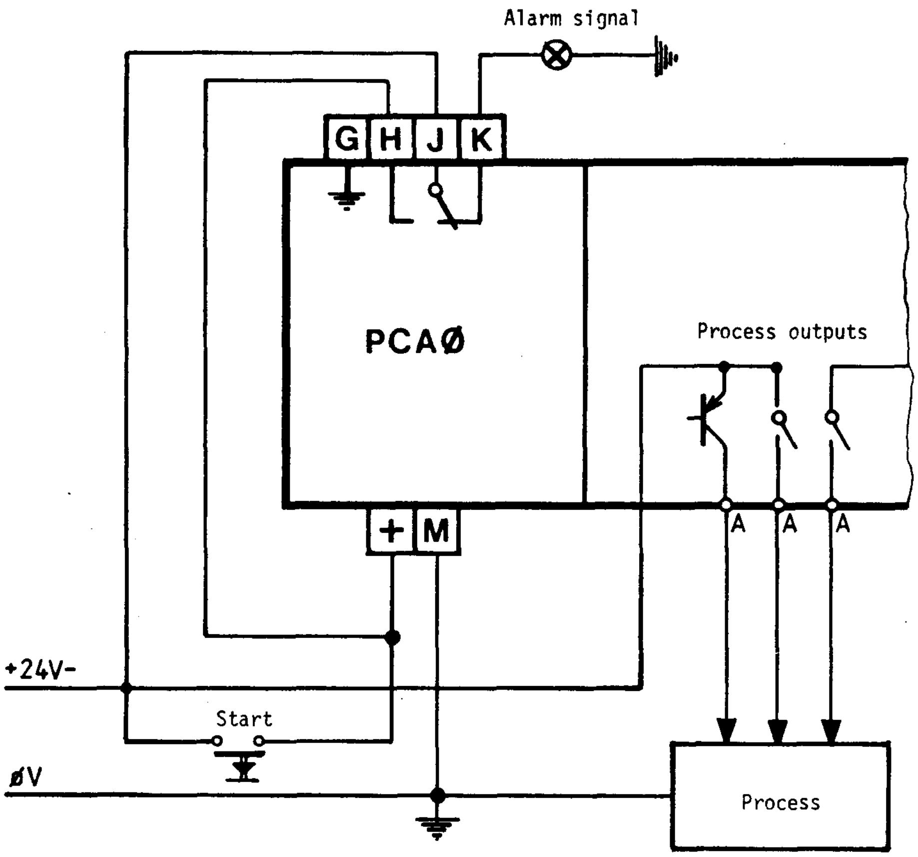
6.3 The WATCHDOG-monitoring circuit

The WD-circuit reliably monitors the correct execution of the user program. In case of an error effective safety measures can be taken.

The WD-relay remains excited (contact H-J is closed) as long as the address 255 receives an alternating signal of ≥ 5Hz. This signal is generated in a circulating program simply with the instruction COO 255. During normal operation of the CPU in the RUN-mode, the terminals H-J remain closed and the green WD-lamp lights up. If a malfunction occurs in the CPU or any other operating mode than "RUN" is selected, the contact H-J opens, the WD-lamp goes out.

For critical systems it is recommended to make use of the WD-function in combination with the following safety circuit. Upon releasing of the WD-relay the PCAØ is no longer supplied with voltage, which has the result that all outputs are reset at once.

Contact rating of the WD-contact: 1.5A, 48VAC or DC



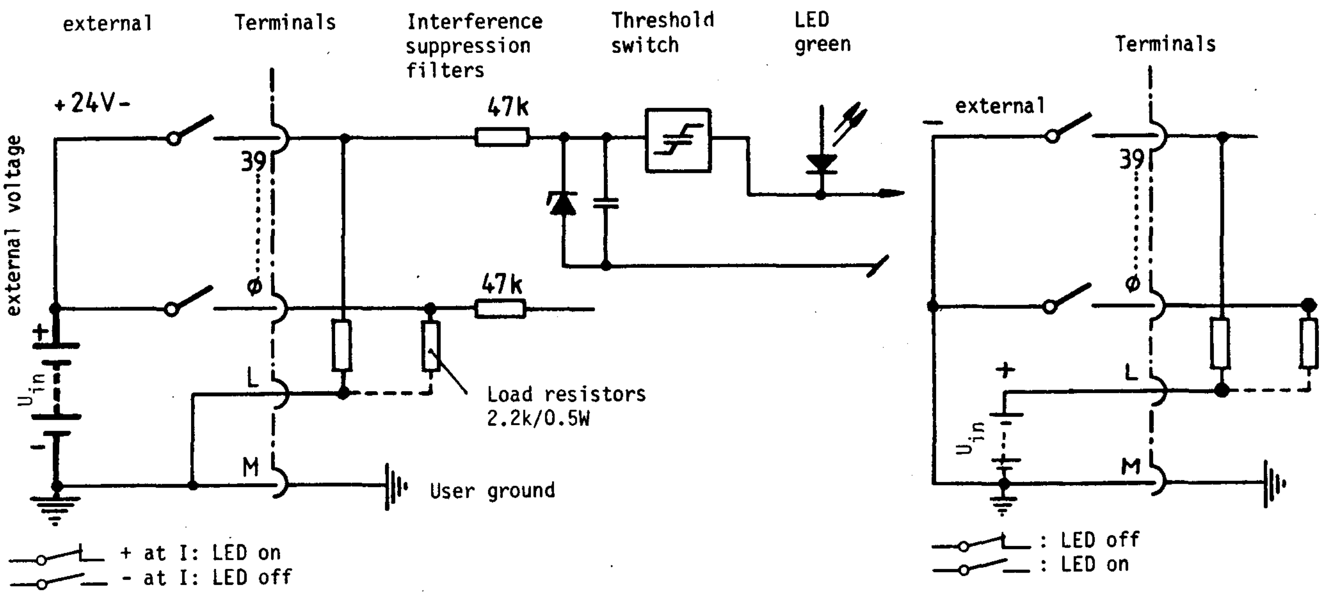
7. Detailed information on inputs and outputs

7.1 Inputs 24VDC (B9Ø)

Input voltage U_{in}	24VDC, smoothed or pulsating
Voltage level	H: 19...32VDC L: Ø... 4VDC
Input current at 24VDC	1ØmA
Input delay	9ms
Operating mode	Source or sink operation, depending on the connection

Source operation

Sink operation



(Terminal assignment)

M	Ø	1	2	3	4	5	6	7	L

---

*) The inputs 16...19 of the PCAØ.M12T can be acted upon only in source operation.

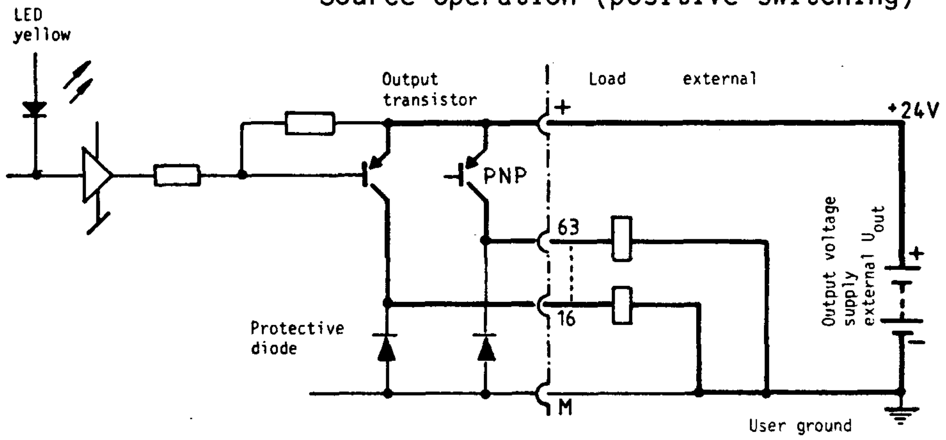
(Terminal assignment)

+	23	22	21	20	19	18	17	16	M

A E

7.2 Transistor outputs (B9∅) Type T

Output current range	5mA...∅.5A In the voltage range 5...24VDC the load resistor should have a value of at least 48 Ohm
Voltage range U_{out}	5...36VDC, smoothed or pulsating
Voltage drop	Max. 1.5V with I = ∅.5A
Operating mode	Source operation (positive switching)

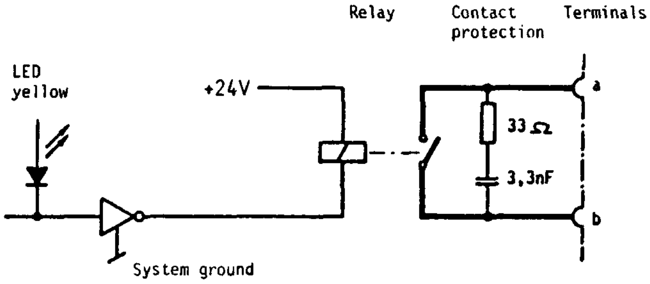


(Terminal assignment)

+	31	30	29	28	27	26	25	24	M

7.3 Relay outputs (A21) Type R

Type of contact	Pure silver, make contact
Contact rating	3A, 25∅VAC AC1 1A, 25∅VAC AC11 3A, 24VDC DC1* 1A, 24VDC DC11*
Contact protection	3.3nF and 33 Ohm
Contact life (AC 1)	3A, 22∅VAC: number of switching cycles is ∅.1 mio. 1.5A, 22∅VAC: number of switching cycles is ∅.5 mio. ∅.3A, 22∅VAC: number of switching cycles is 5 mio.



(Terminal assignment)

a2,7b	a2,6b	a2,5b	a2,4b

*) For reasons of life and reliability transistor outputs should be preferred for switching a DC-voltage of 24V.

8. The operating modes of the PCAØ

In order to prepare and test programs, it is necessary to operate the PC in different operating modes. This is effected by pressing the corresponding keys on the small programming unit PCA2.PØ5 or with the programming interface PCAØ.PØ1 for other programming units.

Please note that these operating mode keys must be actuated for at least Ø.5s for reasons of security. The actual selection of the appropriate operating mode is confirmed by the indicating LED of the programming unit PØ5. When the programming unit connector of the PCAØ is disconnected, the selected operating mode is maintained.

When switching on the PCAØ, the following operating modes are selected automatically:

- With the programming unit connected → STEP (the LED "STEP" on the PØ5 lights up, the green LED "RUN" on the PCAØ does not light up!).
- Without programming unit → RUN (the green LED on the PCAØ lights up).

8.1 Operating modes (summary)

R	RUN	Normal program execution (lamp RUN on the PCAØ lights up)
P	PROG	A user program can be loaded into a RAM-memory (plugged onto the user socket of the PCAØ).
M	MAN	Manual interrogation and setting of elements (inputs, outputs, flags, timers, counters).
S	STEP	Jump to a preselected step address of the user program and step-by-step execution.
B	BREAK	Program execution up to a set "breakpoint" and subsequent step-by-step operation.
T	TEXT	Has no function in the standard PCAØ.

8.2 Detailed description of the operating modes

[R] RUN **Normal program execution**

The PCAØ is automatically in the RUN-mode when switching on if the programming unit is not connected.

[P] PROG **Programming**

A program can be stored in a RAM-memory (on the user socket of the PCAØ) or overwritten (corrected).

Step	Code	Operand
[A] x x x x	[E] x x	x x x x
	[E] x x	x x x x or [C] deleting a wrongly entered line
	....	
	[+] Terminates the input	

Test program [+][+] or [-] [-]

[M] MAN ** **Manual interrogation or setting of elements**

(Elements = inputs, outputs, flags, counters, timers)

Interrogation: [A] $\underbrace{x\ x\ x}_{\text{Element address}}$ (Step) ——► display of the logic state in the operand (Ø/1)

Setting: [A] $\underbrace{x\ x\ x}_{\text{Element address}}$ [E] [1] ◄——► or [Ø]

[S] STEP [+] ——► Display showing where the program is.

Jump to the preselected step address of the user program

[A] 139 [+] ——► Program jumps to step 139, then

[+] [+] ... step-by-step execution of the program with the

result of the logic operation being checkable ✷ ACC = 1*.

Switching to RUN is always possible.

In case of parallel programs, <u>only the activated parallel program</u> is executed in the STEP-mode

---

[B] BREAK **Interruption of the program run and subsequent step-by-step operation**

[+] ——► Display showing where the program is

[+] [+] .... step-by-step execution of the program with the

result of the logic operation being checkable ✷ ACC = 1*.

Switching to RUN is always possible.

In case of parallel programs, all programs are executed simultaneously (as in the RUN-mode).

Setting of a breakpoint

[A] 82Ø [+] ——► program runs up to step 82Ø, then

[+] [+] .... step-by-step operation skipping the "critical" point.

---

*) ACC = accumulator is used to indicate the result of the logic combination. If ACC = 1, (conditions of the logic combination fulfilled), the following switching instructions are executed.

**) If the address of a timer or counter is preceded by a 3 (e.g. 326Ø for counter 26Ø), the value of this register can be read or entered manually with [E] value [+].

9. Programming in three easy steps

C Combined programming incl. arithmetic and index register

B Programming according to flow-chart

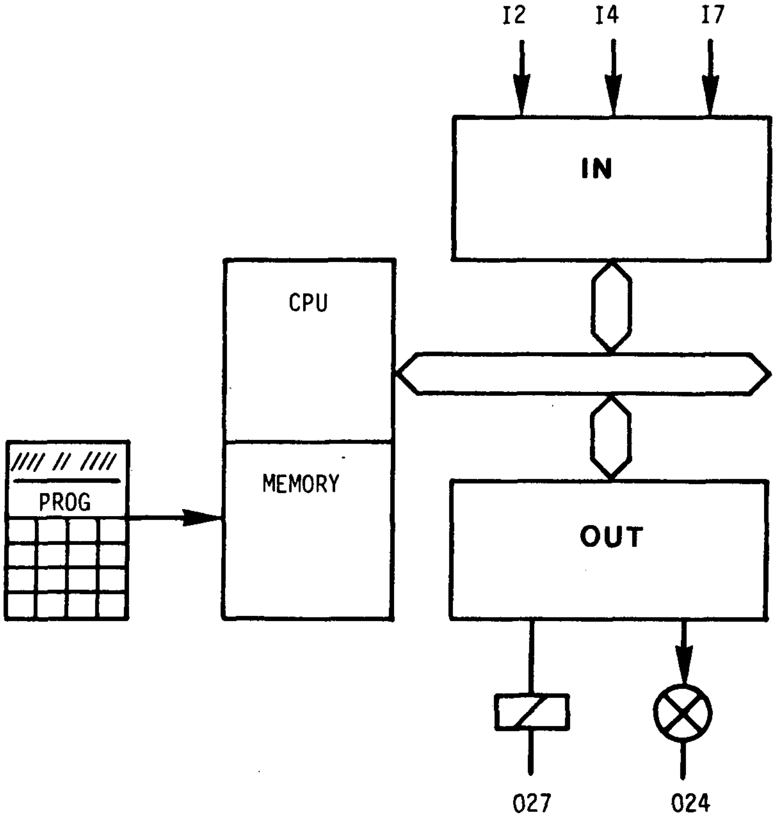
A Programming according to a ladder diagram

The PCAØ is equipped with a very efficient instruction set, level (1H). Thanks to this instruction set even complex control problems can be solved easily. The programs of the PCAØ can be used at any time with other series of the SAIA-PC system family, too. This enables your controller to "grow" according to your requirements, without having to write new programs every time.

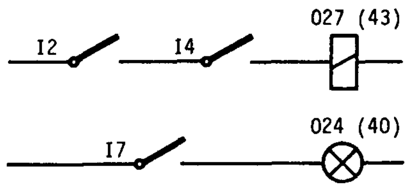
In order to make the start of programming easier for the beginner, the performance of the PCAØ is split up into three easily understandable steps. Maybe your control problem is so easy that it can be solved even on level A.
Let's start programming with a ladder diagram, although this is no longer suitable for modern programming, because process runs are often particularly difficult to represent in relay logic.
But you will see that your PCAØ understands any ladder diagram.

saia®pc

Ⓐ Programming according to ladder diagram



Ladder diagram



Program

```
1   STH   2
2   ANH   4
3   OUT  27
4   STH   7
5   OUT  24
     :
    JMP   1
```

The program prepared according to a ladder diagram is entered into the user memory (RAM) by means of the programming unit. In the RUN-mode, the CPU reads this program line by line and checks the relevant inputs. If one of the contacts I2 or I4 is closed, an "H" (High = voltage greater than +19V) is stored in the CPU and after reading the 2nd line it is AND-connected with the latter. If both contacts are closed (H), the state of the accumulator = 1 (ACCU = 1) and the output 27 is activated, the contactor at output 27 is actuated.

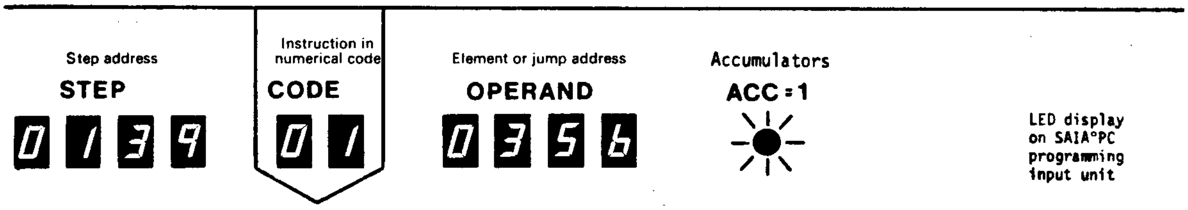
In line 4 the CPU processes input 7. If this contact is closed, output 24 is activated, the indicating lamp lights up.

We can combine more of these logic operations, because our user memory of 4K = 4Ø96 program lines is extremely large. If all logic operations have been programmed, we have to tell the CPU to return to line 1. In this way, our program is permanently run cyclically at a high speed, and all alterations are immediately transferred as logic operation results from the inputs to the outputs.

Instruction set Ⓐ ladder diagram

The instructions available for programming such logic operations are classified into logic instructions and switching instructions.

Their functions are listed in the following table. Do not let yourself be confused by the large number of functions. When programming according to a ladder diagram only about 8 of them are used frequently, which you will soon know by heart.



	Numerical code	Mnemo code	Instruction	Description
Logic Instructions 	Ø1 Ø2	STH STL	Start High Start Low	Start of an operation: Element interrogated for High / Low
	Ø3 Ø4	ANH ANL	And High And Low	And-operation of ACCU with element interrogated for High / Low
	Ø5 Ø6	ORH ORL	Or High Or Low	Or-operation of ACCU with element interrogated for High / Low
	Ø7	XOR	Exclusive Or	Exclusive-or-operation of ACCU with addressed element
	Ø8	NEG	Negate ACCU	Invert state of the ACCU
	Ø9	DYN	Dynamic Control	Signal edge triggering or dynamic control of an operation
Switching Instructions 	1Ø	OUT	Set Output with Status of ACCU	Transfer the state of the ACCU to an output or a flag
	11 12	SEO REO	Set Output Reset Output	Set output or a flag and store Reset output or flag and store
	13	COO	Complement Output	Interrogate state of output or flag and set it to the opposite state
Time Instructions	14	STR	Set Timer -	Set timer to preselected value and start it Time value in 1/10 s (resp. 1/100 s)
Jump Instructions	2Ø	JMP	Unconditional Jump	Unconditional jump to step address
Auxiliary Instructions	ØØ	NOP	No Operation	No operation

The program line

Each instruction in the user program consists of 1 line (in certain cases of 2 lines). In addition to the line number or step address (STEP) a line contains the instruction code (CODE) and operand (OPERAND). The instruction code indicates "WHICH" instruction is to be executed, and the operand determines "WHERE" this instruction is executed.

Structure of the program or instruction line:

STEP	CODE	OPERAND
Ø175	ANH Mnemonic code or Ø3 Numerical code	ØØ63
Line number or step address	Instruction code "WHICH"	Addition to the instruction code "WHERE"

Instruction

Program line

STEP The line no. defines the position of the instruction in the user memory. Decimal numbering from Ø...4Ø95 (4K).

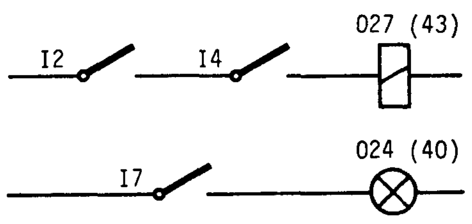
CODE Depending on the programming unit the instruction code can be entered in a 3-digit mnemonic code or in numerical code from Ø to 31. The mnemonic code is based on abbreviations of the corresponding English instructions. It is therefore easy to remember and understood internationally.

OPERAND Here, the address of an element (input, output, timer, counter or flag) or in case of jump instructions the destination address (line no.) is entered.

Timing and counting instructions consist of two lines. In the second line, the appropriate time or counter value appears in the operand.

(A1) A first programming example

Before starting to enter the program, it must be noted down in mnemonic code on paper. For this, it is advantageous to copy the programming lists added at the end of this manual.



Step	Mnemo. code	Numeric. code	Operand	Comment
Ø	NOP	ØØ	Ø	Blank line
1	STH	Ø1	2	Interrogation I2
2	ANH	Ø3	4	AND I4
3	OUT	1Ø	27	Output O27
4	STH	Ø1	7	Interrogation I7
5	OUT	1Ø	24	Output O24
6	JMP	2Ø	1	Return
7				

For programming and simulation we establish the same configuration as described in section 5. All the following examples always refer to the small PCAØ with only 16 or 2Ø inputs. For the bigger versions the output addresses are added in brackets respectively.

With the programming unit PCA2.PØ5 the above program can now be entered into the plug-in user memory (RAM or R95).

[P] ──► operating mode PROG

	Keys	STEP	Mnemonic code	Numerical code	OPERAND	
	[A] [Ø] [E] 1)	ØØØØ	(NOP)	ØØ	ØØØØ	
2)	[E] [Ø] [1] [2]	1	(STH)	Ø1	2	
	[E] [Ø] [3] [4]	2	(ANH)	Ø3	4	
	[E] [1] [Ø] [2] [7]	3	(OUT)	1Ø	27	
	~~[E] [Ø] [1] [8]~~	~~4~~	~~(STH)~~	~~Ø1~~	~~8~~*	3)
3)	[C] [Ø] [1] [7]	4	(STH)	Ø1	7	
	[E] [1] [Ø] [2] [4]	5	(OUT)	1Ø	24	
	[E] [2] [Ø] [1]	6	(JMP)	2Ø	1	
4)	[+]					

1) Selection of the step address Ø ([A] = Address) and erasure of the contents with [E] = Enter.

2) Every following [E] increments the step address by 1 and the contents of the old program line are erased and prepared for the new input.

3) 8 was entered accidently instead of 7. Correction with [C] and repetition of the instruction. The step address is not incremented as a result of [C].

4) The last input must be stored with [+], [E], [A] or [-].

After entering the program it is recommended to compare the program stored in the user memory step by step with the program previously written down:

[A] [1] [+] [+] [+] .....

Now of course we want to test whether the program runs.

[S] ——► The LED indicating operating mode "STEP" lights up

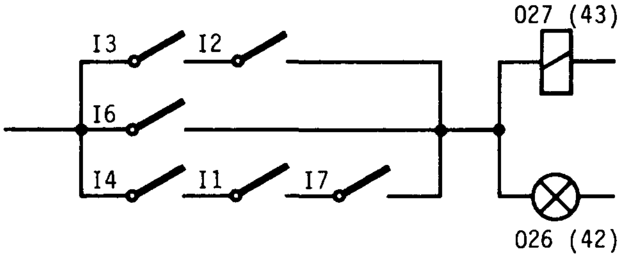
[A] [1] [+] ——► The program execution should start at step 1

[R] ——► The LED "RUN" lights up, the program is running

Now close contacts I2 and I4 ——► the LED of O27 lights up. Upon closing I7 ——► O24 lights up.

(A2) Parallel connection

Ladder diagram



Program

Step	Mnemo. code	Numeric. code	Operand	Comment
1Ø	STH	01	3	Interrogation I3
11	ANH	03	2	AND I2
12	ORH	05	6	OR I6
13	ORH	05	4	OR I4
14	ANH	03	1	AND I1
15	ANH	03	7	AND I7
16	OUT	10	27	Output O27
17	OUT	10	26	Output O26
18	JMP	20	10	Return

Note:

- Every OR-instruction starts a new branch of the parallel connection at the very beginning on the left. Afterwards, AND-operations can be added again to this parallel branch. The whole program, however, is considered as only one logic operation.

- If the logic operations are finished, as many actions as desired (depending on this logic operation) can be added.

- Enter the program as follows:

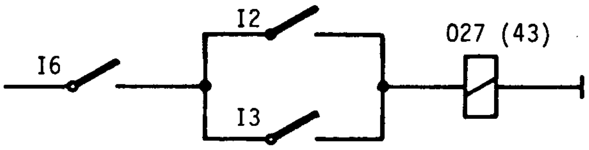
  $\boxed{P}$ ⟶ LED "PROG" for programming mode lights up

  $\boxed{A}$ $\boxed{1}$ $\boxed{Ø}$ $\boxed{E}$ ⟶ The PCAØ is prepared to accept the above program from step address 1Ø

  Continue to proceed as described in example A1, but make the PCAØ execute the program from step 1Ø on with $\boxed{S}$ $\boxed{A}$ $\boxed{1Ø}$ $\boxed{+}$ $\boxed{R}$. Run!

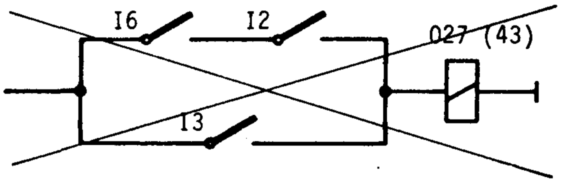
(A3) OR is "stronger" than AND and getting to know other "elements"

The following contact arrangement must be programmed:



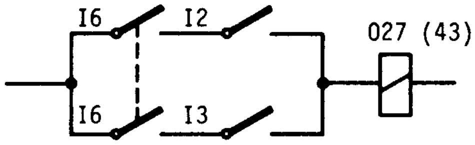
~~STH 6~~
~~ANH 3~~
~~ORH 3~~
~~OUT 27~~

One is tempted to prepare the program as above. However, as OR is always at the beginning of a parallel branch, the following circuit would be the result:



There are two possibilities of implementing the desired function:

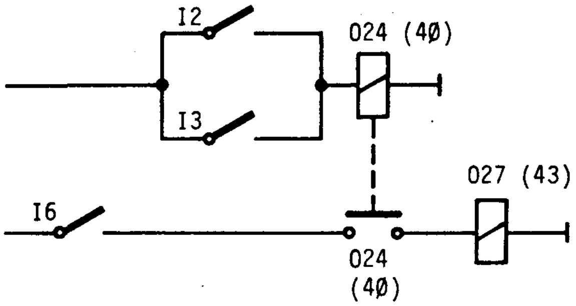
a)



2Ø	STH	Ø1	6
21	ANH	Ø3	2
22	ORH	Ø5	6
23	ANH	Ø3	3
24	OUT	1Ø	27
25	JMP	2Ø	2Ø

The contact I6 is programmed in both parallel branches.

b)



2Ø	STH	2
21	ORH	3
22	OUT	24
23	STH	6
24	ANH	24
25	OUT	27
26	JMP	2Ø

Programming is performed in two steps. As evident from possibility b) outputs can be used as desired in other logic operations, too.

It would be a pity to sacrifice an extra output for this easy task. Therefore, every PCAØ contains 712 FLAGS!

The elements and registers of the PCAØ:

Elements

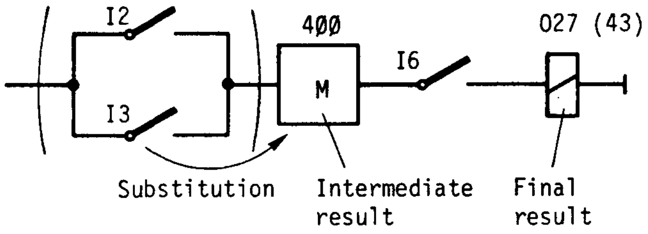
ADDR.	
999 … 765	Retentive flags H or non-volatile flags
764 … 32Ø	Flags M*
319 … 288	32 counters C*, which can also be used as flags
287 … 256	32 timers T* or counters C*
67, 66, 65, 64	4 hardware timers, provided that the module PCAØ.H2Ø is attached
63 … Ø	max. 64 inputs I and outputs O

Register

ADDR.	
319 … 256	64 registers* as counters or timers at 16 bit

*) By inserting the jumper "NVOL" on the CPU all of these locations can be made non-volatile, i.e. when switching off the PCAØ these data are not lost.

With the aid of a flag the ladder diagram can now be drawn as follows:

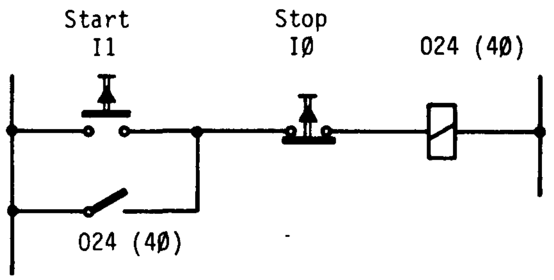


ADDR	NC	MNC	OPRD	
3Ø	Ø1	STH	2	Interrogating I2
31	Ø5	ORH	3	OR I3
32	1Ø	OUT	4ØØ	Storing the intermediate result on flag 4ØØ
33	Ø1	STH	4ØØ	Interrogating flag 4ØØ (and thus the OR-function)
34	Ø3	ANH	6	AND I6
35	1Ø	OUT	27	Output of the result via 027
36	2Ø	JMP	3Ø	Return to the beginning

(A4) Two kinds of start/stop circuit with latching contactor

a) Presented in a ladder diagram

The following classical example is known from the technique of contactors:



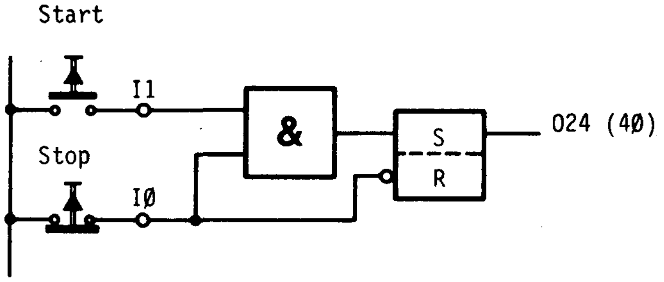
Program

ADDR	NC	MNC	OPRD	
4Ø	Ø1	STH	1	
41	Ø5	ORH	24	Start
42	1Ø	OUT	4Ø1	
43	Ø1	STH	4Ø1	
44	Ø3	ANH	Ø	Stop
45	1Ø	OUT	24	
46	2Ø	JMP	4Ø	

As in example A3 programming is performed using a flag. As evident from the program, the normally open contact IØ is connected with "H", as 024 can be activated only with contact IØ being closed.

This way of programming is fail-safe against wire break. If a wire breaks in the lines of IØ, I1 or 024, 024 is always inhibited.

b) Logic diagram presentation



Program

ADDR	NC	MNC	OPRD	
5Ø	Ø1	STH	1	
51	Ø3	ANH	Ø	Start
52	11	SEO	24	
53	Ø2	STL	Ø	
54	12	REO	24	Stop
55	2Ø	JMP	5Ø	

SEO (11): Set Output With this instruction an element (output or flag) is continuously set until it is reset with REO.

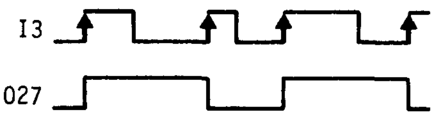
REO (12): Reset Output With SEO/REO we are able to program a flip-flop. Both instructions are executed only if the result of the logic operation is positive (ACCU = 1).

The "Set" instruction of example b) is executed only if IØ = H. If both keys are pressed, the "Reset" instruction takes precedence because of the AND-operation.

This way of programming is also fail-safe against wire break.

(A5) Example of a pulse divider (stepping switch) using the instruction DYN and COO

The following function is referred to as a pulse divider (stepping switch):



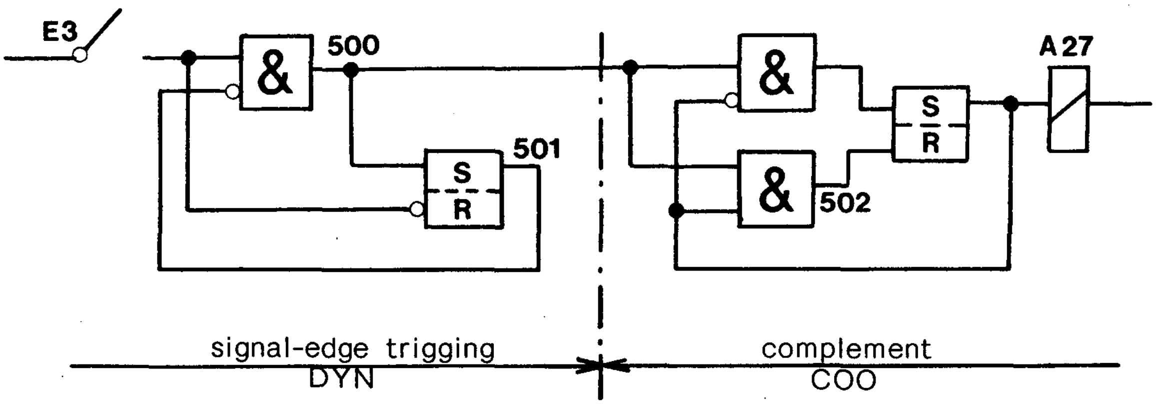
The output 027 is modified only upon each rising edge of I3.

Presenting this function in a ladder diagram results in a quite complicated diagram and a long program.

a) Ladder diagram presentation

Try once on your own to draw a relay diagram that realizes this function. It will not be easy!

b) With SEO/REO and the above function drawn as a logic diagram, it is already much easier.



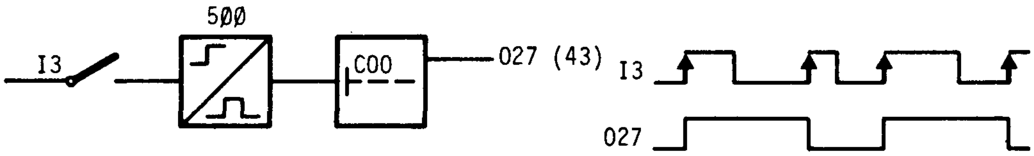
ADDR	NC	MNC	OPRD		
60	01	STH	3	interrogate switches with AND-combination	signal edge triggering (DYN)
61	04	ANL	501		
62	10	OUT	500		
63	11	SEO	501	Flip-Flop	
64	02	STL	3		
65	12	REO	501		
66	01	STH	500	lower AND-combination	Pulse divider or complement output (COO)
67	03	ANH	27		
68	10	OUT	502		
69	01	STH	500	upper AND-combination	
70	04	ANL	27		
71	11	SEO	27	Flip-Flop with output	
72	01	STH	502		
73	12	REO	27		
74	20	JMP	60		

c) With the instructions DYN and COO

DYN (Ø9): Dynamic execution of an interrogation function or signal edge triggering. With the DYN-instruction (together with a flag in the operand) the preceding interrogation instruction accepts only the positive alteration (rising edge). A permanent H-state or the falling edge are ignored.

COO (13): Complement output. The logic state of an output or flag is tested and inverted i.e. if an output is set, it is reset with COO and vice versa.

With these efficient instructions this problem can be solved really easily.



ADDR	NC	MNC	OPRD	
6Ø	Ø1	STH	3	Interrogation of I3
61	Ø9	DYN	5ØØ	Edge triggering, storing in flag 5ØØ
62	13	COO	27	Testing the state of 027 and negating it
63	2Ø	JMP	6Ø	

Without the DYN-instruction in this example the output 027 would be complemented in each program cycle, if switch I3 were closed; that is, approximately 3ØØØ times per second in this short program loop.

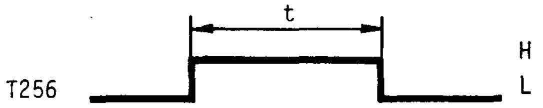
With DYN, output 27 will be complemented only in the 1st program cycle upon closing of I3. Every other cycle will not have any effect, until the signal state of I3 has changed and a new rising edge is formed.

(A6) The software timers with switch-off delay

In example A3 all elements of every standard PCAØ and their corresponding addresses were shown. The 32 software timers reside on the addresses 256...287. For each address there is a 16-bit register in which values from Ø...65535 can be stored.

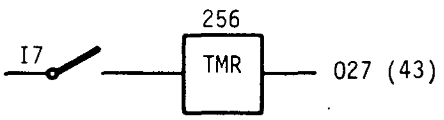
For setting and starting a software timer a two-line instruction is necessary:

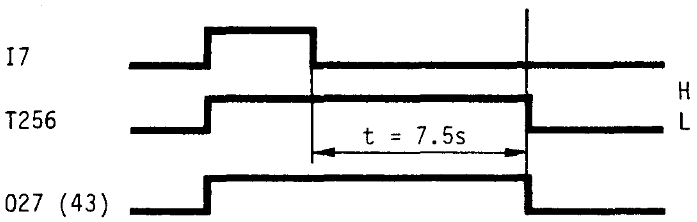
STR (14)	256:	Starting the timer with address 256
ØØ	2Ø:	Loading the corresponding register with the value 2Ø. (Direct input up to 2Ø47 for code ØØ).



When the CPU reads this instruction, the logic state of timer 256 is set to "H" and the register value is reduced according to the time base (1/1Ø s). After this has been done 2Ø times (2Ø x 1/1Ø s = 2 s) the logic state of the timer 256 is "L" again.

With this basic function a switch-off delay can be implemented easily:





ADDR	NC	MNC	OPRD	
7Ø	Ø1	STH	7	Interrogation of I7
71	14	STR	256	Set timer and start } 2-line instruction
72	ØØ	ØØ	75	Input of time in 1/1Øs } 2-line instruction
73	Ø1	STH	256	Interrogation of timer T256
74	1Ø	OUT	27	Transfer to 027
75	2Ø	JMP	7Ø	

Upon closing of I7 the timer is set and its logic state is set to "H". The time does not start to run down before I7 is opened. (In fact, the time starts to run down at once. But as the timer is set again in the next program cycle after some 1ØØµs with I7 closed, the time starts to run down again from the very beginning, until the signal present at I7 is removed, i.e. the contact I7 opens).

If I7 is closed again while the time is running down, the timer is reset and started again.

P.S.: As evident from chapter 5 the time base can be reduced to 1/1ØØs by removing the jumper "1/1Ø" on the CPU thus enhancing the resolution.

(A7) Use of the hardware timer module PCAØ.H2Ø

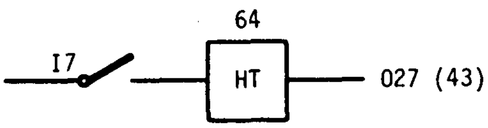
In addition to the 32 precise software timers (which are included in every standard version) 4 hardware timers (additional module) are available in order to use time functions in the programs.

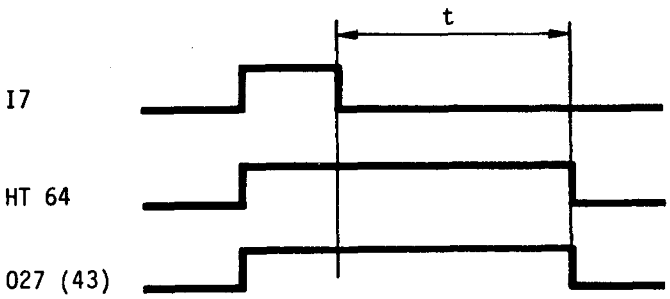
The 4 hardware timers with the addresses 64...67 can be operated in a similar way to the software timers.

Software timers		Hardware timers	
STR	256	REO	64
ØØ	t	SEO	64

For the hardware timer the time range and time are set directly on the hardware module PCAØ.H2Ø.

Example according to A6:






```
ADDR  NC  MNC  OPRD
 8Ø   Ø1  STH     7    Interrogation of I7
 81   12  REO    64  } Start hardware timer HT 64
 82   11  SEO    64  }
- - - - - - - - - - -
 83   Ø1  STH    64    Interrogation of HT 64
 84   1Ø  OUT    27    Transfer to 027
 85   2Ø  JMP    8Ø
```

As with the software timer, the time starts to run down at once in case of the HT 64, too. As long as I7 is switched on however, HT 64 is reset in each program cycle. The time actually starts to run down when I7 is opened resulting in the above switch-off delay.

(A8) Software timers with fleeting-on delay with external time input in BCD-code

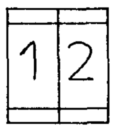
The PCAØ also allows reading the BCD-values directly into the registers of the timers and counters. As a result, time values can be changed at any time with the BCD-switch. If we use the input simulation unit PCA2.SØ5 as described in chapter 5, we can see that the input addresses I8...I15 are acted upon by a two-digit BCD-switch. For this, the BCD-switch transmits the following signals to the input:

Binary signals present at 4 inputs (BCD-switch)				
I 12	I 13	I 14	I 15	Decimal value
$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	
L	L	L	L	Ø
L	L	L	H	1
L	L	H	L	2
L	L	H	H	3
L	H	L	L	4
L	H	L	H	5
L	H	H	L	6
L	H	H	H	7
H	L	L	L	8
H	L	L	H	9

Now, the STR-instruction can expanded in such a way that the delay time can be directly read off the BCD-switch.

Code	OPERAND	
STR (14)	256	1st line
NC	En	2nd line

BCD-switch
(2 pieces)



I8...15
En

En : The highest input address of the two BCD-switches (e.g. 15) is introduced into the operand.

NC : The numerical code has the values 16, 17 or 18 with the following meanings:

16 value Ø... 99 x time base = Ø...9.9s*
or 17 value Ø... 99Ø x time base = Ø... 99s*
or 18 value Ø...99ØØ x time base = Ø...99Øs*

---

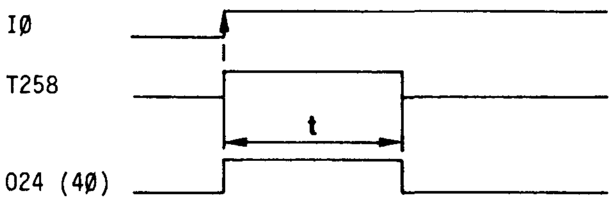
*) These values result from the standard clock rate of 1/1Øs. It can be changed to 1/1ØØs by removing the jumper "1/1Ø".

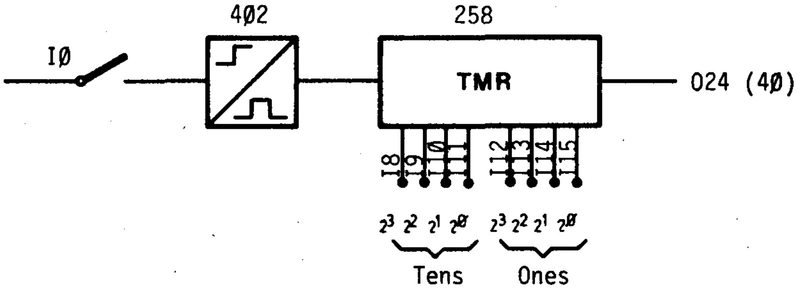
saia pc

Problem:

The time range 1 to 99s must be set with the external BCD-switch. The inputs I8...15 are acted upon by the two BCD-switches. The time function is to have a fleeting-on delay.

Because of the **DYN**-instruction only the rising edge of IØ is taken into account, enabling the timer 258 to run down without interruption.





Program:

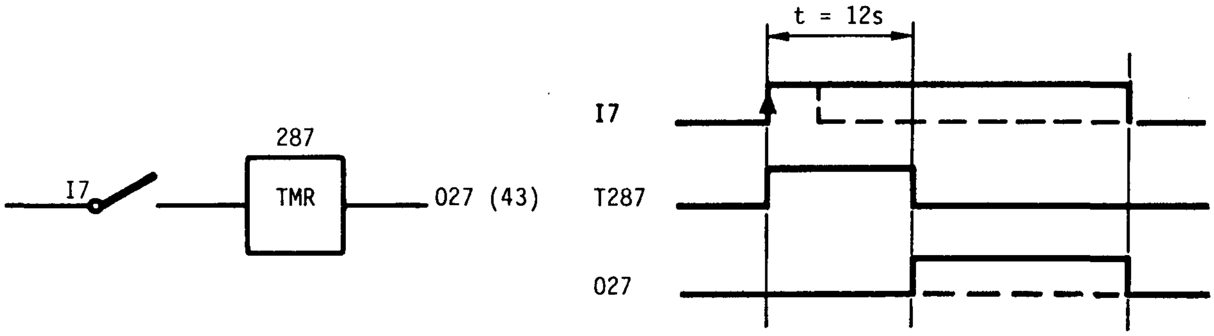
ADDR	NC	MNC	OPRD	
9Ø	Ø1	STH	Ø	Interrogation of IØ
91	Ø9	DYN	4Ø2	Edge triggering, timer is set only in the 1st cycle
92	14	STR	258	Start timer and set it to the external value
93	17	17	15	x 1Ø x 1/1Øs using the inputs 8...15
94	Ø1	STH	258	Interrogation of timer
95	1Ø	OUT	24	Transfer to 024
96	2Ø	JMP	9Ø	

If the same function has to be performed by a hardware timer, just replace the lines 92 and 93 as follows:

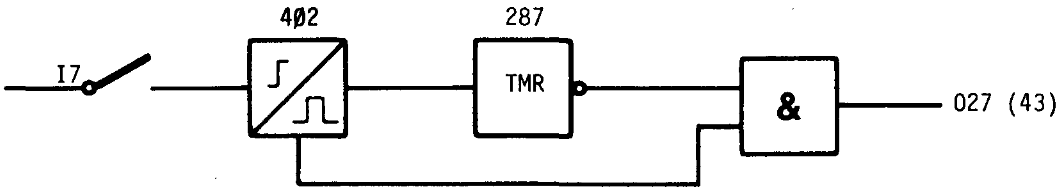
ADDR	NC	MNC	OPRD	
92	12	REO	66	Start hardware timer HT 66
93	11	SEO	66	
94	Ø1	STH	66	Interrogation of hardware timer HT 66

(A9) Switch-on delay with two new, very useful instructions

Problem:



In case of the switch-on delay the timer is also started by the rising edge of I7 as in case of the fleeting-on delay. However, output 27 should be switched on only after the time has elapsed.
order to activate 027, the rising edge of I7 must have been generated and the timer run down (L). This can be shown by means of the following logic diagram:



As we are working with an intelligent PC, elapsing of the time has to be displayed on the programming unit. This is achieved with the instruction **DTC (31): "Display Timer or Counter"**. Provided that this instruction is executed at least once every second (which does not constitute a problem in circulating programs), the actual contents of the corresponding timer or counter are displayed in the operand field of the PØ5. Activation of the DTC is effective only if the ACCU = 1. Therefore, it needs to be preceded by the instruction **SEA (19) Ø: "Set Accumulator"**.

Program:

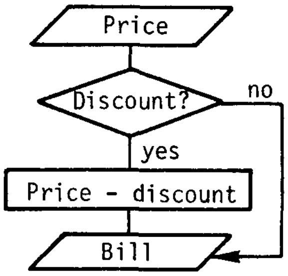
ADDR	NC	MNC	OPRD	
1ØØ	Ø1	STH	7	Interrogation of I7
1Ø1	Ø9	DYN	4Ø2	Signal edge triggering
1Ø2	14	STR	287	Start timer and set it to
1Ø3	ØØ	ØØ	12Ø	12Ø x Ø.1s = 12s
1Ø4	Ø1	STH	4Ø2	Interrogation of the edge flag of I7
1Ø5	Ø4	ANL	287	AND timer run down (L)
1Ø6	1Ø	OUT	27	then output to 027
1Ø7	19	SEA	Ø	Set ACCU = 1
1Ø8	31	DTC	287	Display of the timer contents
1Ø9	2Ø	JMP	1ØØ	

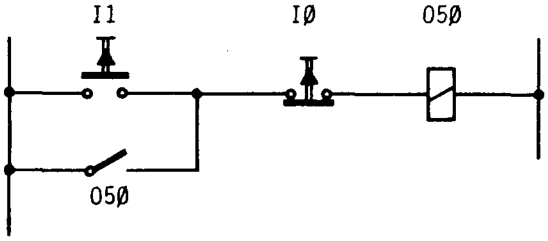
If this function is to be performed by the hardware timer HT 67, the lines 1Ø2/1Ø3 and 1Ø6 must be replaced as follows:

ADDR	NC	MNC	OPRD	
1Ø2	12	REO	67	Start hardware timer HT 67
1Ø3	11	SEO	67	
.			.	
.			.	
1Ø6	Ø4	ANL	67	AND HT 67 run down (L)

Ⓑ Programming according to a flow-chart

Show a ladder diagram and this flow-chart to a 12-year old pupil. What do you think will he interpret?

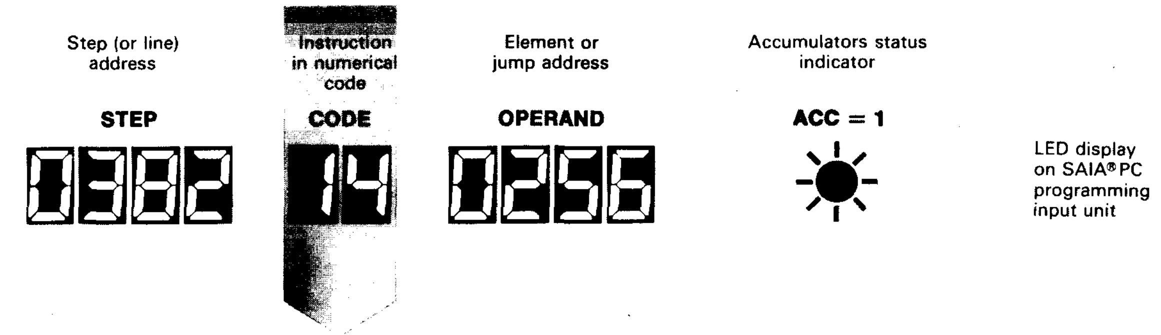


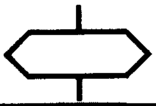
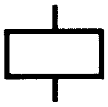


So, is it surprising that more and more industrial processes, too, are being described using a flow-chart. Especially in the chemical or food-processing industry, as well as in many branches of mechanical engineering, there are many processes that can be described in an easy and understandable way with the aid of a flow-chart.

The PCAØ comprises many instructions which can be used efficiently for programming according to a flow-chart resulting in much easier and clearer functions.

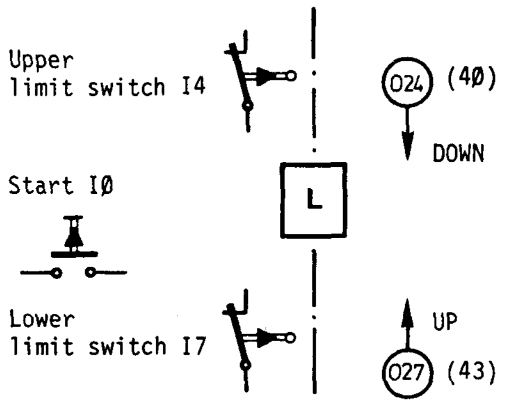
Counter instructions can be used of course both for programming with the aid of a ladder diagram and flow-chart. In order to prevent level Ⓐ from getting too complex, we have assigned them to level Ⓑ.



	Numerical code	Mnemo-code	Instruction	Description
Timing and Counting Instructions	15*	SCR*	Set Counter	Set counter to determined value (if accu = 1)
	17 18	INC DEC	Increment Counter Decrement Counter	Increment / Decrement } content of the counter by 1 (if accu = 1)
Jump Instructions	2Ø	JMP	Unconditional Jump	Unconditional jump to step address
	21 22	JIO JIZ	Jump if Accu is One Jump if Accu is Zero	Jump if { accu = 1 / accu = Ø } to step address
	[illegible] 24	JMS RET	Jump to Subroutine Return from Subrout.	Jump to subroutine (regardless of accu) Return from subroutine (regardless of accu)
Wait Instructions	25 26	WIH WIL	Wait if High Wait if Low	Wait whilst interrogated element is { High / Low
Auxiliary Instructions	ØØ	NOP	No Operation	No operation
	19	SEA	Set Accu	Set accumulator status to 1
	29**	PAS**	Program Assignment	Assignment of the parallel programme
	[illegible]	DOP	Display Operand	Display content of operand (if accu = Ø)
	[illegible]	DTC	Display Timer or Counter	Display timer or counter value (if accu = 1)

* Two line instruction (second line contains determined value)
** Two line instruction (second line contains starting address of parallel programme)

(B1) Upwards/downwards movement



Problem:

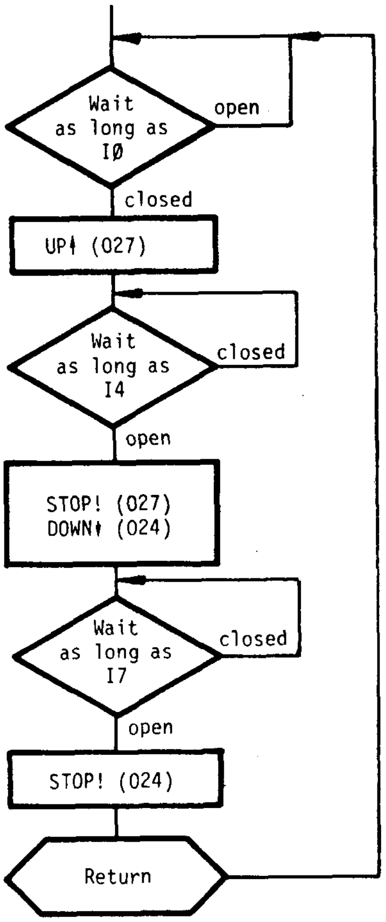
As a result of a pulse at IØ the load L must be moved upwards (UP = 027) until I4 is opened. Then, the load must be lowered again (DOWN = 024) until I7 is opened.

Additional conditions:

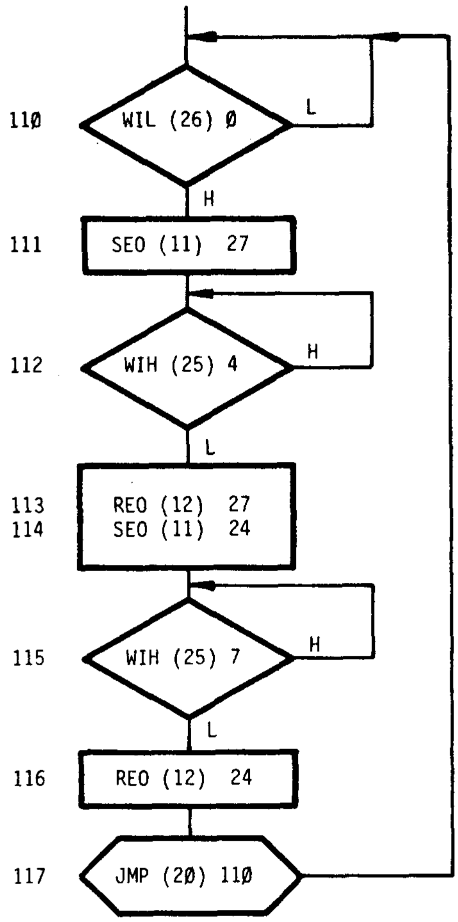
- If IØ is permanently on this sequence must be continuously repeated.
- After a voltage break an upwards movement must not be triggered before a pulse is present at IØ.
- Note: Before starting the program the switches I4 and I7 must be closed.

Problem solving by means of a flow-chart:

Flow-chart



Program



Comment

Wait as long as IØ is open (low).
If IØ is closed, then ...

Set output 27 (UP)

Wait as long as I4 is closed (high).
If I4 is opened, then ...

Reset 027 (STOP) and set 024 (DOWN)

Wait as long as I7 is closed (high).
If I7 is opened, then ...

Reset 024 (STOP)

Return to the beginning

As evident from the example, a sequence is described step-by-step when programming according to a flow-chart. The stages of the process are split up into conditions (wait for an input state) and actions (set or reset outputs).

If we follow this program in step-by-step operation in the operating mode STEP (key sequence ([S] [A] 11Ø [+] [+])...., we will notice that the processor itself stays in the wait loop, until the condition for continuing is fulfilled. This means: the program does not permanently circulate cyclically like the program prepared with the aid of a ladder diagram, but the program is executed according to the progress of the process.

This has three important advantages:

- The programs are made clearer, because they show the individual steps of a process and not an abstraction in a ladder diagram which is not related to the process.

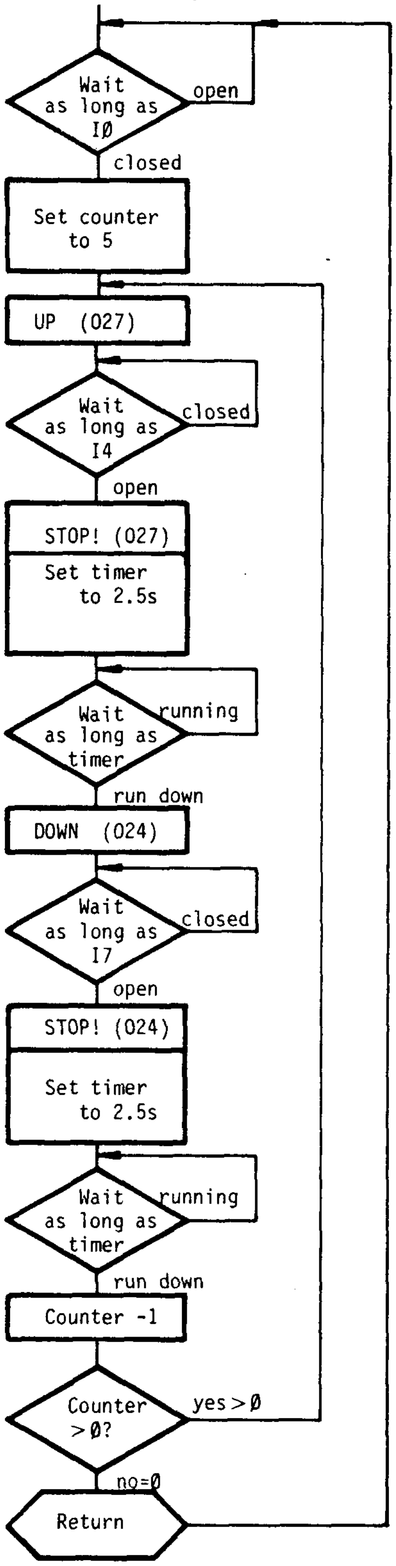
- Only those program sections are processed which are of importance for the current stage of the process. As a result, possible malfunctions resulting from the execution of irrelevant program parts are avoided. Most importantly, however, the reaction time between step condition and action is considerably reduced.

- If the slide L stops after the first up-/down-movement, it is possible to exactly localize the error in the program by means of the programming unit in the operating mode "STEP", [+] [+] ... At step 11Ø, WIL Ø is displayed. Moreover, the LED of IØ indicates that this input has not been activated. Therefore, the error can be quickly narrowed down to the contact or the connecting line of IØ.

(B2) Up-/downwards movement with timer and counter

Problem:
The movement described in example B1 is to be effected not only once but a certain number of times (e.g. 5 times) and a pause of 2.5s must be made each time at the points where the direction is reversed.

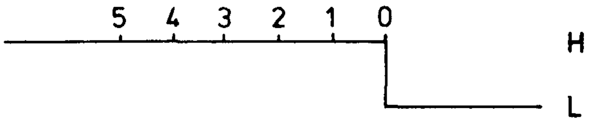
Solution:
Flow-chart



Line	Programm	Comment
130	WIL (26) Ø — L: wait / H: continue	Wait, as long as IØ is open (low). If IØ is closed, then ...
131	SCR (15) 3ØØ	Set counter 3ØØ to 5
132	ØØ 5	
133	SEO (11) 27	Set output 27 (UP)
134	WIH (25) 4 — H: wait / L: continue	Wait as long as I4 is closed (high). If I4 is opened, then ...
135	REO (12) 27	Reset 027 (STOP).
136	STR (14) 256	Set timer 256 to 2.5s
137	ØØ 25	
138	WIH (25) 256 — H: wait / L: continue	Wait while timer is running down (high)
139	SEO (11) 24	Set output 24 (DOWN)
140	WIH (25) 7 — H: wait / L: continue	Wait as long as I7 is closed (high). If I7 is opened, then ...
141	REO (12) 24	Reset 024 (STOP)
142	STR (14) 256	Set timer 256 to 2.5s
143	ØØ 25	
144	WIH (25) 256 — H: wait / L: continue	Wait, while timer is running down (high)
145	DEC (18) 3ØØ	Decrement counter 3ØØ by 1
146	STH (Ø1) 3ØØ	Test counter, if Ø (high)
147	JIO (21) 133	If yes, jump
148	JMP (2Ø) 13Ø	Return to the beginning

Contrary to problem B1, in this example timers were also integrated into the action parts. One must wait for these to run down in additional conditional loops (WIH 256).

The use of the instruction **SCR (15): Set counter** is new.
This is a two-line instruction (as in case of the timer). In our example the counter is set to 5. As long as the counter C3ØØ is greater than zero, its logic state is "H".



With the instruction **DEC (18): Decrement Counter** the counter state is decremented by 1 in each run.

The interrogation instruction STH 3ØØ allows to continuously check the logic state of the counter. If the ACCU = 1 after the interrogation, the counter contents are still greater than zero and another loop must be executed. Returning to the beginning of a loop is effected with the conditional jump instruction **JIO (21): Jump if ACCU = 1**.

In other words: If the counter is greater than zero, another loop must be executed: If the counter state is zero, no more jumps are effected and the last JMP-instruction leads back to the program beginning.

If the hardware timer HT 65 is used instead of software timer T256, the corresponding program parts need to be replaced as follows:

```
STR  256 }        { REO 65
 ØØ   25 } ---->  { SEO 65
WIH  256 }        { WIH 65
```

(B3) What needs to be done, if we want other functions to be performed together with the upwards/downwards movement?

In this case, we prepare a second or third parallel program out of the 16 parallel programs provided for each PCAØ-standard version!

Problem:

Simultaneously and independently of program B2 a blinker with a flashing time of Ø.3s has to be generated in a parallel program and the actual counter state of the upwards/downwards movement has to be continuously displayed on the programming unit.

Program:

```
 12Ø  PAS  (29)   ①  } The parallel program number ① is assigned to
  21   ØØ       13Ø  } the up/down program starting with step address 13Ø
  22
┌►123  DTC  (31)  3ØØ    The counter state of counter 3ØØ           ┐
│                        must be indicated continuously             │
│124  STH            2 ┐                                            │
│ 25  ANL          257 │                                            │
│ 26  STR          257 │ Output 16 blinks every Ø.3s                ├  Parallel
│ 27   ØØ            3 │ as soon as I2 is closed                    │  program Ø
│ 28  COO           16 ┘                                            ┘
└129  JMP          123
```

In the very beginning, we use the two-line instruction **PAS (29) 1 (program assignment)** to assign the number ① to the parallel program starting at step address 13Ø in the so-called assignment part.

Other parallel programs can be added easily with PAS ② or PAS ③.

The program at step addresses 123 to 129 is a so-called circulating program without wait loops. It may include other functions such as monitoring or other permanently performed tasks. This parallel program without assignment is automatically given the number Ø.

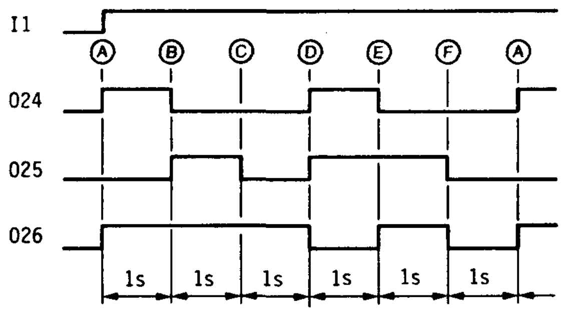
If you want to execute PPØ at the same time from step address 123 on and PP1 from step address 13Ø on, then jump to the start address 12Ø with the following assignment: [S] [A] 12Ø [+] [R].

Now, test both programs! Both are executed separately and asynchronously. As mentioned above, up to 16 parallel programs of any desired length can be executed with the PCAØ (as with all SAIA-PCs).

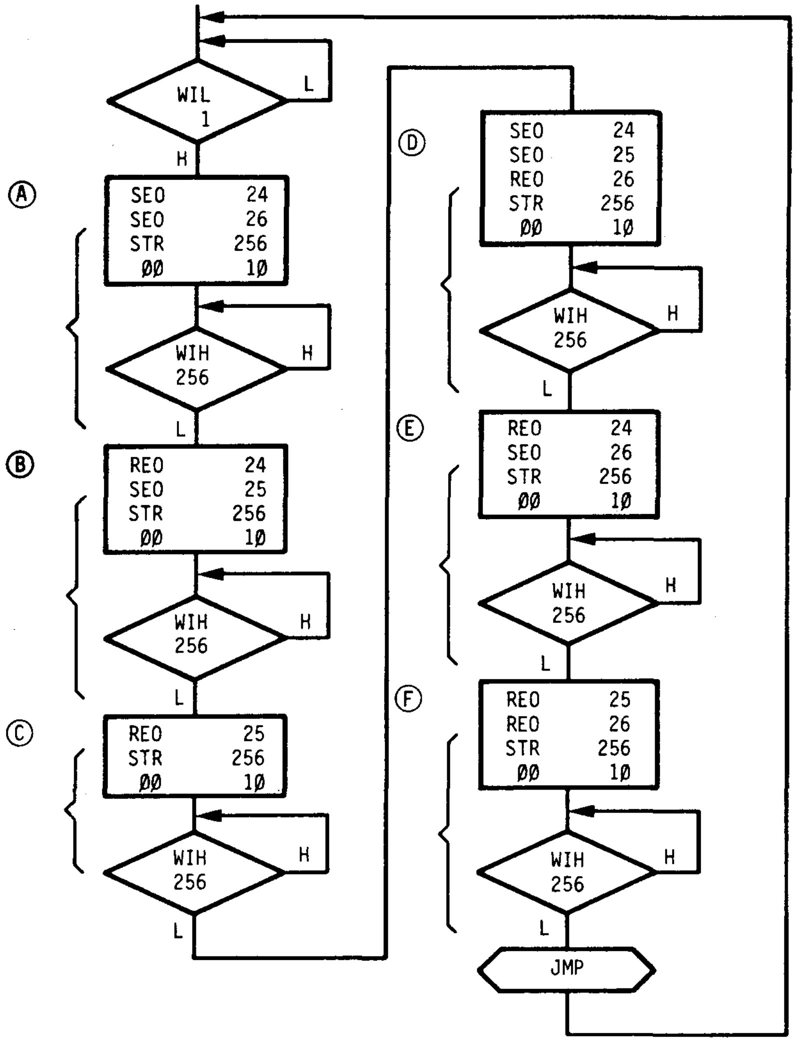
(B4) Subroutines help you save much time and result in shorter and clearer programs

Problem:

The following sequence has to be executed after closing of I1:



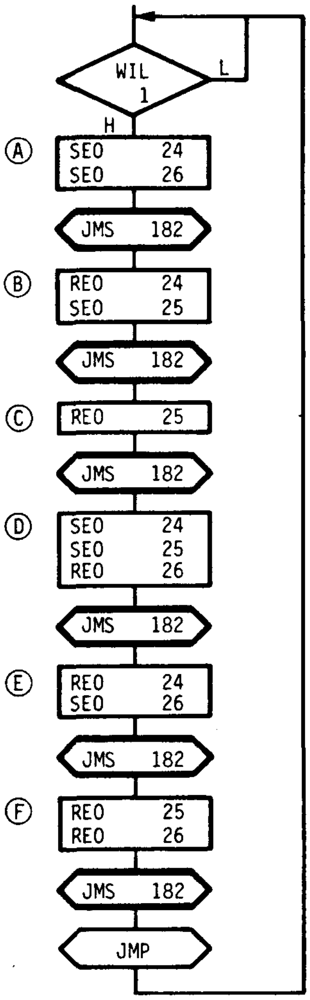
Solution a: Of course with the aid of a flow-chart (if you want, try to prepare the program with the aid of a ladder diagram!).



The flow-chart shows the program sequence without subroutines. The program parts which are marked with the brackets are repeated 6 times. Therefore, it is of advantage to write them as subroutines.

Solution b: with subroutine

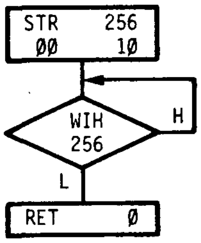
Main program




```
******************* Main program
ADDR  NC  MNC  OPRD
 16Ø  26  WIL     1    Wait, if I1 is open
 161  11  SEO    24
 162  11  SEO    26
 163  23  JMS   182 ⇒Jump to subroutine 182
 164  12  REO    24
 165  11  SEO    25
 166  23  JMS   182 ⇒Jump to subroutine 182
 167  12  REO    25
 168  23  JMS   182 ⇒Jump to subroutine 182
 169  11  SEO    24
 17Ø  11  SEO    25
 171  12  REO    26
 172  23  JMS   182 ⇒Jump to subroutine 182
 173  12  REO    24
 174  11  SEO    26
 175  23  JMS   182 ⇒Jump to subroutine 182
 176  12  REO    25
 177  12  REO    26
 178  23  JMS   182 ⇒Jump to subroutine 182
 179  2Ø  JMP   16Ø
```

Thanks to the instruction **JMS (23)**: Jump to Subroutine the same program parts must be prepared only once. Every subroutine ends with the instruction **RET (24)**: Return with the operand Ø.

Subroutine "182"




```
=================== Subroutine 182 (wait 1s)

⇒182  14  STR   256
 183  ØØ   ØØ    1Ø
 184  25  WIH   256
 185  24  RET     Ø
```

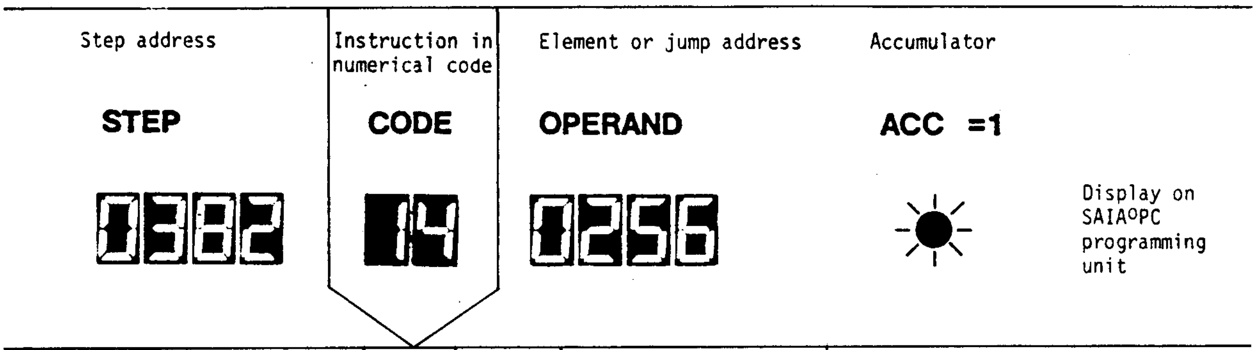
It is also possible to program the subroutine of the subroutine of the subroutine i.e. down to the 3rd level.

(C) Indexing, arithmetic and check sum

At the programming level C you are already a professional!

Certainly, one can live happily without climbing up to this level. But, once you are up here, you will be proud of yourself and the PCAØ for having prepared your programs in such a nice and efficient way.

These are the remaining instructions for the software level (1H) of the SAIA-PC system family.

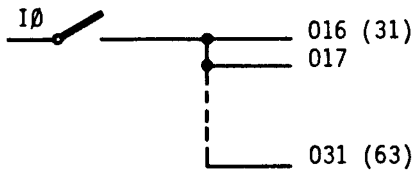


	Num. code	Mnemo. code	Instruction	Description
Transfer instructions	15	SCR	Set counter	
		19	2nd line	Read-in 5x4 bit BCD
		2Ø	2nd line	Output 5x4 bit BCD
		21	2nd line	Output 8 bit binary
		22	2nd line	Output 12 bit binary
		23	2nd line	Output 16 bit binary
		24	2nd line	Read-in 8 bit binary
		25	2nd line	Read-in 12 bit binary
		26	2nd line	Read-in 16 bit binary
		31	2nd line	Transfer counter to counter resp. IR
Arithmetic instructions	15	SCR	Set counter	
		27	2nd line	Addition +
		28	2nd line	Subtraction -
		29	2nd line	Multiplication x
		3Ø	2nd line	Division :
Indexing instructions	16	SEI	Set index	Set index register to preselected value
	27	INI	Incr. index	Increment the index reg. by 1
	28	DEI	Decr. index	Decrement the index reg. by 1
			OPERAND	
Special instructions	29	PAS	18	Changing the number of active parallel programs
These instructions consist of two lines	29	PAS	3Ø...34	Check sum

(C1) In case of short programs address indexing has considerable effects

Problem:

All outputs of the PCAØ are to be activated upon closing of input IØ.



Without address indexing this program would consist of 34 steps for a PCAØ with 32 outputs.

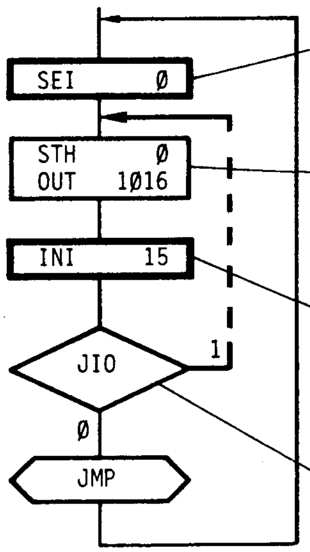
Solution:

With indexing, however, the program will always consist of only 6 lines, irrespective of the number of outputs.

ADDR	NC	MNC	OPRD	
2ØØ	16	SEI	Ø	Set index register (IR) to starting value Ø
2Ø1	Ø1	STH	Ø	Interrogation of input Ø
2Ø2	1Ø	OUT	1Ø16	Set indexed outputs
2Ø3	27	INI	15	Incrementing of IR up to the final value 15
2Ø4	21	JIO	2Ø1	Return, as long as IR < 15
2Ø5	2Ø	JMP	2ØØ	——►

In the following flow-chart the individual functions are easier to understand:

Indexing loop



With the instruction SEI (16) the index register is set to Ø (max. value is 255).

1ØØØ is added to the first output address 16. As a result, this indexed instruction is processed by the CPU as follows:
1Ø16 - 1ØØØ + IR

As a result of instruction **INI** (27) the index register is incremented by 1 in each run until it reaches the value 15.

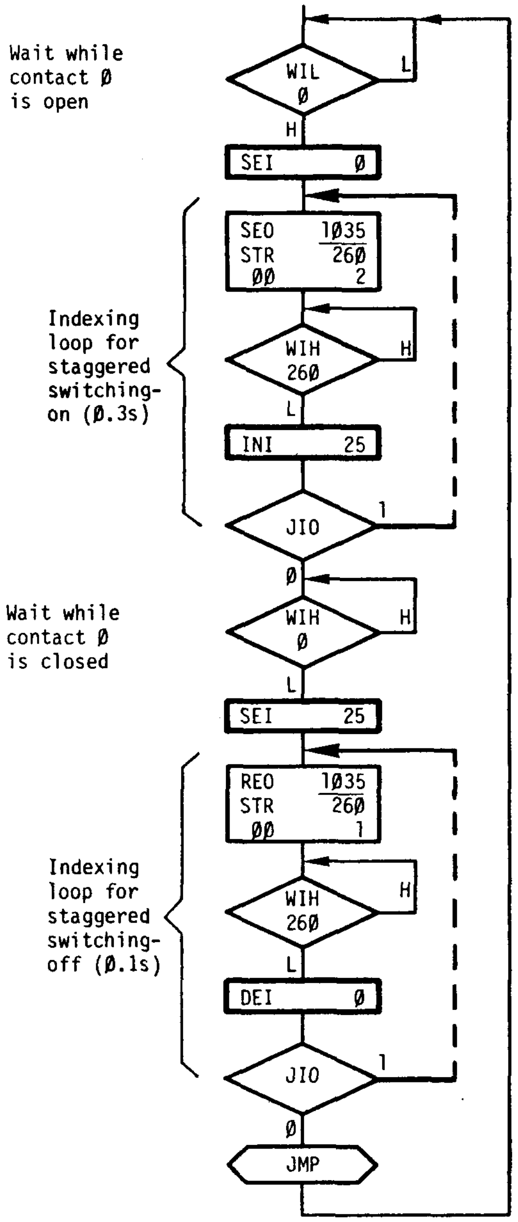
As long as IR < 15, the ACCU = 1, which causes the return.
If IR = 15, the indexing loop is left. All outputs from 16...31 have been processed.

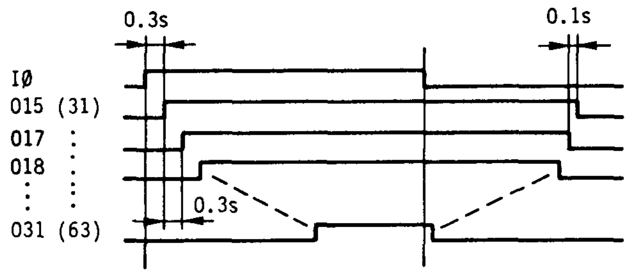
Ⓒ2 Thanks to upward and downward indexing high flexibility is obtained

Problem:

Upon closing of IØ the outputs O16...31 (63) are to switch on one after the other every Ø.3s. Upon opening of IØ the outputs O31 (63)...16 should be switched off again in reverse order every Ø.1s.

Solution:





ADDR	NC	MNC	OPRD	
				TURN-ON PHASE
21Ø	26	WIL	Ø	
11	16	SEI	Ø	
212	11	SEO	1Ø16	016 + 1ØØØ = 1Ø16
13	14	STR	26Ø	
14		ØØ	3	
15	25	WIH	26Ø	
16	27	INI	15	031 - 16 = 15
217	21	JIO	212	
				TURN-OFF PHASE
218	25	WIH	Ø	
19	16	SEI	15	031 - 16 = 15
22Ø	12	REO	1Ø16	016 + 1ØØØ = 1Ø16
21	14	STR	26Ø	
22		ØØ	1	
23	25	WIH	26Ø	
24	28	DEI	Ø	
25	21	JIO	22Ø	
226	2Ø	JMP	21Ø	→

The upper indexing loop is very similar to that in example C1. In the lower loop the index register is set to 15 first. In the first run REO 1Ø16 - 1ØØØ + 15 = REO 31 acts upon output 31. The instruction **DEI (28)** decrements the value of the IR by 1, as a result of which output 3Ø is switched off in the next run, etc. As soon as the IR = Ø, no further return is effected, the indexing loop is left.

You will have noticed that in case of the version with relay outputs, a pause is made after 4 relays respectively. This is due to the fact that the 4 unoccupied addresses 2Ø...23 are processed too.

saia®pc

(C3) Even calculating is possible with the little PCAØ

The 64 counter registers of the PCAØ can be used in many ways. In connection with the instructions **STR** and **SCR** we have only dealt with the codes ØØ and 16, 17, 18 of the second line. As evident from the following table, 32 functions are available.

With the codes Ø1...15 the area from 2Ø48 to 65535 can be reached.

With the codes 19...26 8- to 2Ø-bit digital values can be loaded into the counter register or transferred to elements.

Finally, with the codes 27...3Ø arithmetic functions can be performed.

Code 31 serves to transmit values from index registers to a counter register or from one counter register to another.

	Mnemonic code	Numerical code	Operand	Explanations
1. line	STR/SCR	14/15	256...319	Address of the register
2. line		ØØ	xxxx	Value in the operand + Ø
		Ø1		+ 2Ø48
		Ø2		+ 4Ø96
		Ø3		+ 6144
		Ø4		+ 8192
		Ø5		+ 1Ø24Ø
		Ø6		+ 12288
		Ø7		+ 14336
		Ø8		+ 16384
		Ø9		+ 18432
		1Ø		+ 2Ø48Ø
		11		+ 22528
		12		+ 24576
		13		+ 26624
		14		Value in the + 28672
		15	xxxx	operand + 3Ø72Ø
		16	Highest addr. of 8 subsequent elements	2 x 4 bit BCD x 1
		17		2 x 4 bit BCD x 1Ø
		18		2 x 4 bit BCD x 1ØØ
		19	Highest addr. of the sequence of elements	Read instr. for 5 x 4 bit BCD
		2Ø		Output instr. for 5 x 4 bit BCD
		21		Output instr. for 8 bit binary
		22		Output instr. for 12 bit binary
		23		Output instr. for 16 bit binary
		24		Read instr. for 8 bit binary
		25		Read instr. for 12 bit binary
		26		Read instr. for 16 bit binary
		27	xxx	Addition } with a constant (Ø...255) or with the contents of a T/C (256...319)
		28	xxx	Subtraction
		29	xxx	Multiplication
		3Ø	xxx	Division
		31	iii	iii = Ø: value of index register is loaded into counter iii = 256...319: Value of corresponding T/C is loaded into counter

Problem:

In order to introduce the arithmetical possibilities of the PCAØ, we will play a bit with figures in this example. Processing of numerical values plays an important role in counting problems or when analog values have to be processed.

The following functions are assigned to the inputs of our simulation unit:

IØ	:	Addition	+
I1	:	Subtraction	-
I2	:	Multiplication	x
I3	:	Division	:

I7	:	Storing of the 1st value via BCD-preselection switch
I6	:	Storing of the 2nd value via BCD-preselection switch
I5	:	Triggering of the arithmetic operation
I8...15	:	BCD-preselection switch
C26Ø	:	Register for first value
C27Ø	:	Register for second value
C266	:	Display register for DTC
024 (56)	:	Acknowledgement for storing of 1st value (I7)
025 (57)	:	Acknowledgement for storing of 2nd value (I6)
026 (58)	:	Acknowledgement for performing the operation

The individual values stored with I7 and I6 have to be indicated on the operand display. This applies also to the result of the arithmetic operation preselected with IØ...I3.

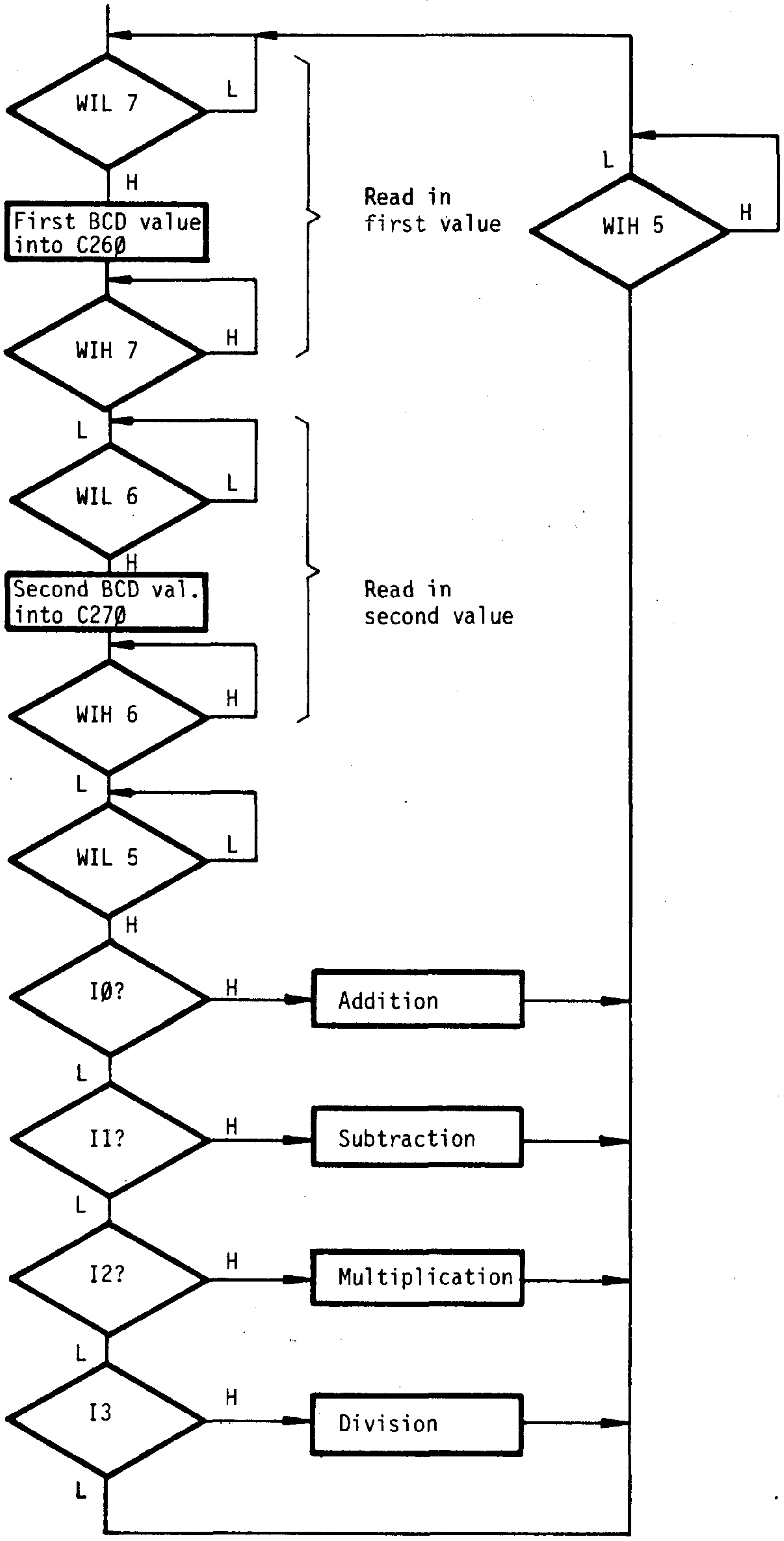
Example: $\boxed{87 - 25 = 62}$

- . Turn on I1 ⟶ subtraction
- . Set 87 via BCD-switch
- . Upon switching on and off I7 the 1st value is loaded into counter C26Ø
- . 87 is indicated on the operand display, at the same time LED 24 lights up acknowledging the first step
- . Set 25 via BCD-switch
- . Upon switching on and off I6 the second value is loaded into counter C27Ø
- . 25 is indicated on the operand display, at the same time LED 25 lights up acknowledging the second step
- . When I5 is on: operation is being performed
- . The result of 62 appears on the operand display, at the same time LED 26 lights up acknowledging the third step
- . Upon switching off I5 the outputs 24, 25, 26 are reset, with I7 a new input can be effected
- . If the subtraction had a negative result (2nd figure > 1st figure), 9999 appears on the display
- . If in a division a value is divided by Ø, 8888 is displayed. In both cases, one benefits from the fact that the CPU sets the ACCU = Ø in these special cases

Solution:

We use a flow-chart, which enables us to follow the individual steps of the process.

Rough flow-chart



Program:

In order to display the numerical values continuously, we execute DTC 266 in the parallel circulating program PPØ, while the flow-chart containing PP1 is assigned to address 238.

```
 ADDR  NC  MNC  OPRD
  23Ø  29  PAS     1  } Assignment of PP1 from
  231       ØØ   238  } address 238 on
 -------------------
  232  ØØ  NOP     Ø
  233  ØØ  NOP     Ø
 ===================
┌►234  31  DTC   266  } PPØ display C266
└─235  2Ø  JMP   234  }
 ===================
┌►238  26  WIL     7    PP1                            ┐
   39  11  SEO    24    Acknowledgement of step 1     │
   4Ø  15  SCR   26Ø    BCD value to C26Ø             │ Storing the
   41       16    15                                   > 1st value
   42  15  SCR   266 )  Copy C26Ø to C266             │ (2-digit)
   43       31   26Ø )  for display                   │
   44  25  WIH     7                                   ┘
 -------------------
  246  26  WIL     6                                   ┐
   47  11  SEO    25    Acknowledgement of step 2     │
   48  15  SCR   27Ø    BCD value to C27Ø             │ Storing the
   49       16    15                                   > 2nd value
   5Ø  15  SCR   266 )  Copy C27Ø to C266             │ (2-digit)
   51       31   27Ø )  for display                   │
   52  25  WIH     6                                   ┘
 -------------------
  254  26  WIL     5                                   ┐
   55  11  SEO    26    Acknowledgement of step 3     │
   56  Ø1  STH     Ø                                   │
   57  21  JIO   265  ───► Jump for addition          │
 -------------------                                   │
   58  Ø1  STH     1                                   │
   59  21  JIO   27Ø  ───► Jump for subtraction       > Performing
 -------------------                                   │ operation
   6Ø  Ø1  STH     2                                   │
   61  21  JIO   28Ø  ───► Jump for multiplication    │
 -------------------                                   │
   62  Ø1  STH     3                                   │
   63  21  JIO   285  ───► Jump for division          ┘
 -------------------
┌─264  2Ø  JMP   295
▼
```

```
ADDR  NC  MNC   OPRD
 265  15  SCR   260    C260 + C270 ——> C260                  Addition
  66       27   270
  67  15  SCR   266    Copy C260 to C266
  68       31   260    for display
  69  20  JMP   295
--------------------
 270  15  SCR   260    C260 - C270 ——> C260                  Subtraction
  71       28   270
  72  22  JIZ   276    Jump, if result is negative
  73  15  SCR   266    Copy C260 to C266
  74       31   260    for display
  75  20  JMP   295
 276  15  SCR   266    Load display counter
  77       00   9999*  with 9999
  78  20  JMP   295
--------------------
 280  15  SCR   260    C260 x C270 ——> C260                  Multiplication
  81       29   270
  82  15  SCR   266    Copying 260 to C266
  83       31   260    for display
  84  20  JMP   295
--------------------
 285  15  SCR   260    C260 : C270 ——> C260                  Division
  86       30   270
  87  22  JIZ   291    Jump, if division by 0
  88  15  SCR   266    Copy 260 to C266
  89       31   260    for display
  90  20  JMP   295
 291  15  SCR   266    Load display counter
  92       00   8888*  with 8888
--------------------
 295  25  WIH     5    Step of operation finished?
  96  12  REO    24
  97  12  REO    25    Resetting the acknowledgement outputs
  98  12  REO    26
 299  20  JMP   238 ——> Return to the beginning of PP1
--------------------
```

Manual interrogation or loading of a counter register:
If you want to check the contents of a counter register or modify these manually at any time, proceed as follows:

E.g. display C270

[M] Press key "MAN" 0.5s

[A] 3270 (that is, 3000 is added to the counter address 270)

——> Counter contents are displayed

---

*) As the operand can be at most 2047, enter the following (according to the table in example C3):
instead of 00 9999 ——> 04 1807 (4 x 2048 + 1807)
instead of 00 8888 ——> 04 696 (4 x 2048 + 696)

E.g. enter the value 1234 into C266

[M] Press key "MAN" Ø.5s

[A] 3266 (address + 3ØØØ)

[E] Ø 1234 (values < 1Ø'ØØØ must be preceded by Ø)

[+]

If contact I7 is open, the value introduced in the above program is displayed with DTC, too, in the RUN-mode.

It is hard to believe how many possibilities the little PCAØ offers!

(C4) If long jumps are to be programmed

A program consisting of 2Ø47 steps is a long program for a PCAØ. But the user memory has a capacity of Ø...4Ø95 steps.

If long jumps with end addresses in the second half of the user memory, i.e. from 2Ø48 to 4Ø95 are to be programmed, these jump instructions must be entered using two lines.

This applies to all jump instructions JMP, JIO, JIZ and JMS.

a) Jump instructions with operands 1 to 2Ø47
(Operand ØØØØ is not allowed, see b.)

Example: Conditional jump with end address 1845

```
either   | JIO (21)   1845 |  ——> one line as usual

or       | JIO (21)      Ø |  } two lines, with the first
         |       ØØ   1845 |  } line containing the operand Ø
```

b) Jump instructions with operands 2Ø48 to 4Ø95

Example: Jump to the subroutine 328Ø

```
For programming:  5Ø1  | JMS (23)      Ø |   [E]
                  5Ø2  |       ØØ   328Ø |   [E]

For checking:     5Ø1    JMS (23)      Ø     [+]
                  5Ø2          Ø1   1232     [C] ≙ convert
                  5Ø2          EE   328Ø     [+]
```

The value Ø1 1232 residing in the user memory corresponds to the jump address 328Ø. Ø1 stands for the multiple of 2Ø48 and 1232 is the remainder (1 x 2Ø48 + 1232 = 328Ø).

With key [C] the actual jump address is displayed, with the code containing the character EE (applies to the programming unit PCA2.PØ5).

In case of a jump instruction with the operand Ø the second line is automatically read for the end address. Therefore, a jump to the address Ø always consists of two lines:

```
| JMP (2Ø)     Ø |
|       ØØ     Ø |
```

c) The following is an example using a blinker in a subroutine. The subroutine starts at address 35ØØ.

```
Main program                          Subroutine

ADDR  NC   MNC   OPRD                 ADDR  NC   MNC   OPRD
 5ØØ  26   WIL      1            =>   35ØØ  Ø2   STL    256
 5Ø1 |23   JMS      Ø|  }             35Ø1  14   STR    256
 5Ø2 |       ØØ  35ØØ|  }  =>         35Ø2         ØØ     2
 5Ø3  2Ø   JMP    5ØØ                 35Ø3  13   COO     24
                                      35Ø4  24   RET      Ø ——>
```

Ⓒ5 The instruction **PAS 18** serves to stop the parallel programs, if they are no longer required

PAS 18 — Limitation of the assigned parallel programs

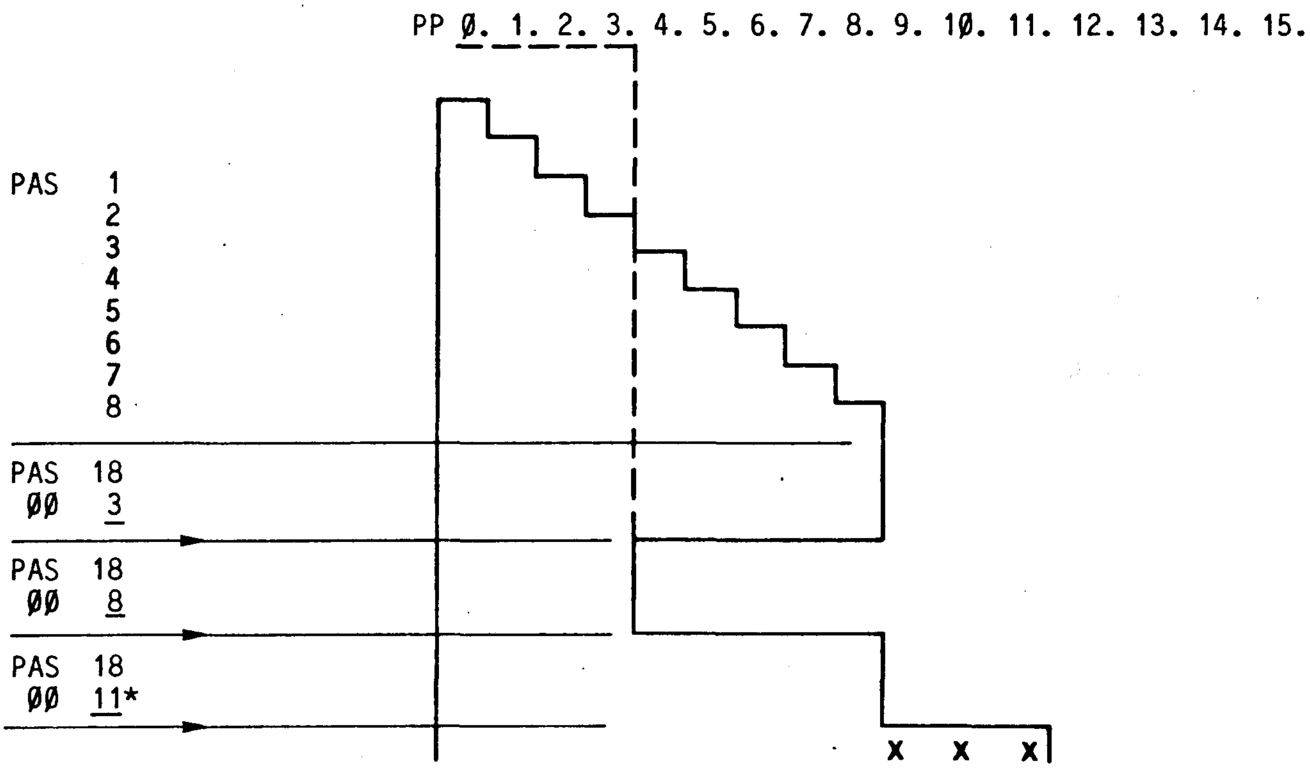
All SAIA-PCs allow assignment of up to 16 parallel programs and running them in parallel. Up to now it has been necessary to reassign a PP to a dummy program loop if it was no longer required. However, no time could thus be saved during the execution of the remaining PPs.

In case of the PCAØ, however, it is now possible to limit a maximum number of active PPs with the PAS 18-instruction.

After the PP-assignment with PAS Ø...max. 15 a limited number of PPs can be determined in the user program at any place and as many times as desired.

1st line	PAS 18	
2nd line	ØØ x	; x in the range Ø...15

Parallel program assignment — Assigned PP



*) If a higher number than the PPs originally assigned is entered with PAS 18, no malfunction is caused. The inactive PPs 9...11, however, will require processing time in the system program.

(C6) With the instruction "Check-sum" the reliability of your PCAØ is considerably increased

PAS 3Ø and PAS 31...34 "Check Sum" of the system and user program

The "Check Sum" serves to establish the check sum of the memory contents of system programs (PAS 3Ø) or of user programs or texts (PAS 31...34). Thus, it can be ensured that the contents of the memories checked have not been changed.

After execution of the instruction:

- ACCU = 1 If the comparison is correct
- ACCU = Ø If the reference value does not comply with the check sum.

The instructions PAS 3Ø...34 are always executed irrespective of the ACCU state.

If a change in contents has occurred, the user can take the measures which seem necessary to him: triggering an alarm, resetting the watchdog etc.

```
PAS   3Ø        ; Check sum of the system program
 ØØ    Ø        ; 2nd line is always ØØ  Ø

PAS   31...34   ; Check sum of the user program, 31...34 being intro-
                  duced corresponding to the program sections 1K...4K.
 xx   xxxx      ; Reference value
```

The appropriate reference value for the user program is obtained by executing the respective PAS-instruction in the operating mode STEP. The PCAØ displays this check sum on the PCAØ programming unit for a few seconds. In the operating mode PROG, the corresponding reference value can then be introduced in the 2nd line of the PAS 31...34 instruction.

Attention: Execution of these instructions takes quite a long time:

PAS 3Ø = 28.Øms
PAS 31...34 = 8.3ms

Therefore, use "Check Sum" only if the sequence to be controlled allows it: e.g. when switching on the PC, at the end of an operation cycle, etc.

It is recommended not to introduce this instruction into the user program until it has been completely developed and tested. Each program alteration, irrespective of whether the program was extended or reduced, results in an alteration of the "Check Sum".

Example: A 2K-user program is to be executed upon switching on

```
2Ø   PAS    31  } Check Sum
21    Ø9   825  } 1.K
22   JIZ    35
23   PAS    32  } Check Sum
24    Ø7  154Ø  } 2.K
25   JIZ    35

3Ø   COO   255  } Main program with
31   JMP    3Ø  } WD-monitoring

35   SEO    16  } Set alarm output
36   JMP    35  } outside the main
                  program
```

Proceed:
- After entering and checking the program, select operating mode STEP
- Type in ADR 23 +
  → the reference value for 2.K (PAS 32) appears for approx. 2Ø sec.
- Type ADR 24 (+) for input of reference value in mode PRG
- Same procedure for PAS 31

ANLAGE:

Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				

Mnemocode	Zahlencode	Befehl wird nur aus-geführt wenn ACCU =	Befehl setzt ACCU = 1	ACCU wird beeinflusst
NOP	ØØ			
STH	Ø1			X
STL	Ø2			X
ANH	Ø3			X
ANL	Ø4			X
ORH	Ø5			X
ORL	Ø6			X
XOR	Ø7			X
NEG	Ø8			X
DYN	Ø9			X
OUT	1Ø			
SEO	11	1		
REO	12	1		
COO	13	1		
STR	14	1		
SCR	15	1		
SEI	16		X	X
INC	17	1		
DEC	18	1		
SEA	19		X	X
JMP	2Ø		X	X
JIO	21		X	X
JIZ	22		X	X
JMS	23		X	X
RET	24		X	X
WIH	25		X	X
WIL	26		X	X
INI	27			X
DEI	28			X
PAS	29			
DOP	3Ø	Ø		
DTC	31	1		

Ein-, Ausgänge	Ø - 255
Zeitglieder, Zähler	256 - 287
Merker	288 - 764
Haftspeicher	765 - 999

ANLAGE:

Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar	Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
ØØ					5Ø				
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				
1Ø					6Ø				
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				
2Ø					7Ø				
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				
3Ø					8Ø				
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				
4Ø					9Ø				
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				

Logik-Befehle									
ØØ	Ø1	Ø2	Ø3	Ø4	Ø5	Ø6	Ø7	Ø8	Ø9
NOP	STH	STL	ANH	ANL	ORH	ORL	XOR	NEG	DYN

Aktions-Befehle									
1Ø	11	12	13	14	15	16	17	18	19
OUT	SEO	REO	COO	STR	SCR	SEI	INC	DEC	SEA

Sprung-Befehle						
2Ø	21	22	23	24	25	26
JMP	JIO	JIZ	JMS	RET	WIH	WIL

Hilfs-Befehle				
27	28	29	3Ø	31
INI	DEI	PAS	DOP	DIC

Eingänge Ausgänge	Ø-255
Zeitglieder Zähler	256-287
Merker	288-764
Haftspeicher	765-999

SAIA AG
CH-3280 Murten Switzerland

Industrial Components and Controls

Belgique
Landis & Gyr Belge SA, Dépt. Industrie
Avenue des anciens combattants 190, B-1140 Bruxelles
☎ 02/244 02 11, Tx 65 930

Danmark
E. Friis-Mikkelsen A/S
Krogshøjvej 51, DK-2880 Bagsvaerd
☎ 02/98 63 33, Tx 37 350

Deutschland
Landis & Gyr GmbH
Friesstrasse 20–24, Postfach 600 529, D-6000 Frankfurt 60
☎ 069/4 00 20, Tx 0417 164

España
Landis & Gyr BC SA
Batalla del Salado 25, Apartado 575, Madrid 7
☎ 91/467 19 00, Tx 22 976

France
Acir Sàrl
29–31, rue de Naples, F-75008 Paris
☎ 1/782 37 95, Tx 630 680

Great Britain
Landis & Gyr Ltd
Elgee Works, Victoria Road, North Acton, London W3 6XS
☎ 01/992 53 11, Tx 21 486

Italia
Landis & Gyr SpA, Divisione Commerciale
Via P. Rondoni 1, I-20146 Milano
☎ 02/42 48, Tx 332 142

Nederland
Landis & Gyr BV, Div. Electrowater
Kampenringweg 45, Postbus 444, NL-2800 AK-Gouda
☎ 01820-65 432, Tx 20 657

Norge
Malthe Winje & Co A/S
Cort Adelers gate 14, Postboks 2440, Solli, N-0202 Oslo 2
☎ 02/56 59 90, Tx 19 629

Österreich
Landis & Gyr Gesellschaft m. b. H
Breitenfurter Strasse 148 a, Postfach 9, A-1230 Wien
☎ 0222/84 26 26-0, Tx 132 706

Schweiz
Saia AG, Verkauf Schweiz
CH-3280 Murten
☎ 037/72 11 85, Tx 942 127

Suomi
Finnland
OY Landis & Gyr AB
SF-02430 Masala
☎ (90) 29731, Tx 121 037

Sverige
Beving Elektronik AB
Box 21 104, S-10031 Stockholm
☎ 08/15 17 80, Tx 10 040

Australia
Landis & Gyr (Australia) Pty Ltd
411 Ferntree Gully Road, P. O. Box 202, Mount Waverley, Vic. 3149
☎ 3/544-2322, Tx 32 224

Printed in Switzerland 26/79 E2 3.87 H07