System Description

ABB Procontic K200
Family of Compact
Programmable Controllers

Software

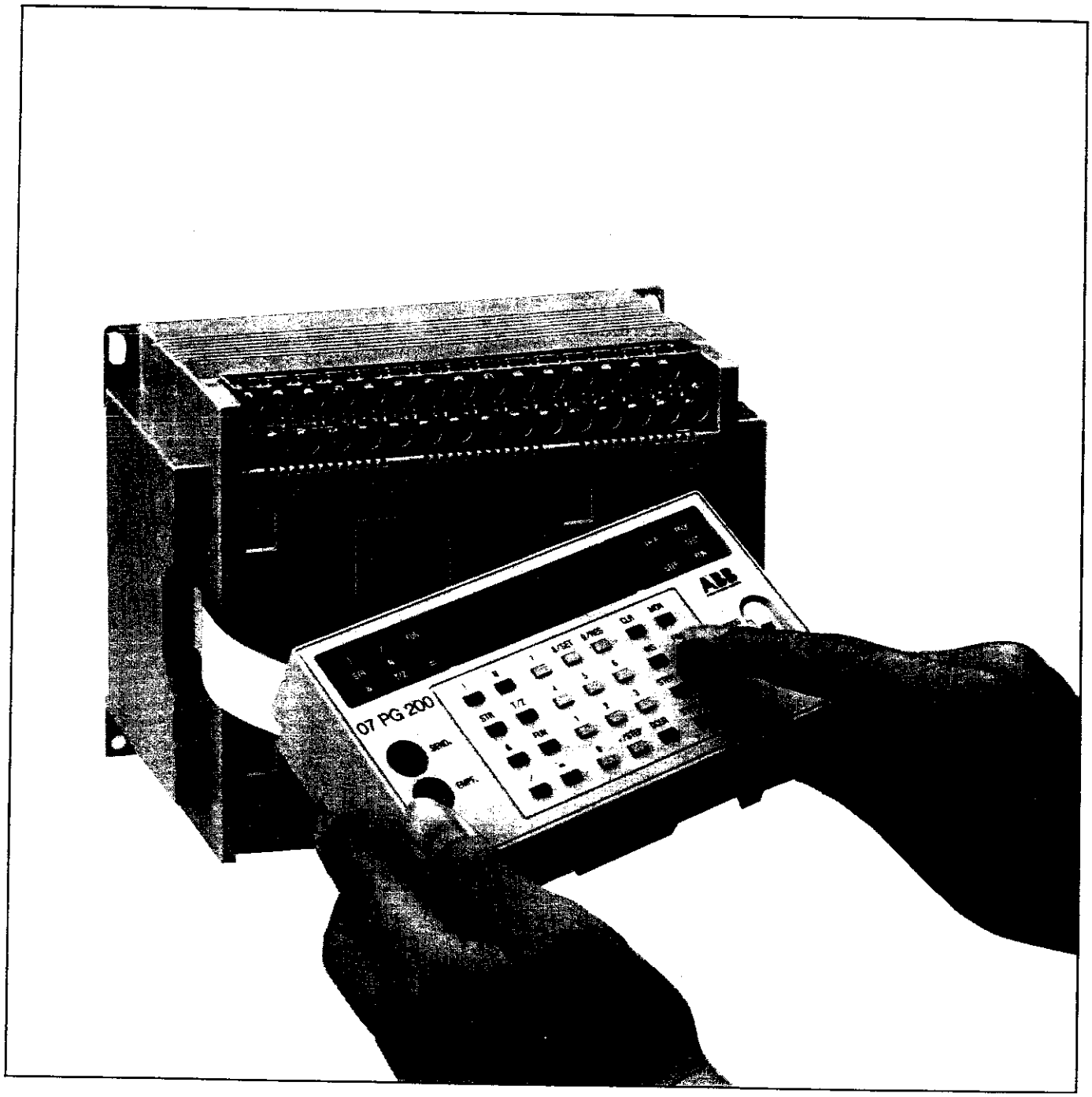


**ABB Schalt-
und Steuerungstechnik**

ABB

Regulations
Concerning the Setting up of Installations

Apart from the basic "Regulations for the Setting up of Power Installations" DIN VDE* 0100 and for "The Rating of Creepage Distances and Clearances" DIN VDE 0110 Part 1 and Part 2 the regulations "The Equipment of Power Installations with Electrical Components" DIN VDE 0160 in conjunction with DIN VDE 0660 Part 500 have to be taken into due consideration.

Further attention has to be paid to DIN VDE 0113 Part 1 and Part 200 in case of the control of working and processing machines. If operating elements are to be mounted near parts with dangerous contact voltage DIN VDE 0106 Part 100 is additionally relevant.

If the protection against direct contact according to DIN VDE 0160 is required, this has to be ensured by the user (e.g. by incorporating the elements in a switchgear cabinet). The devices are designed for pollution severity 2 in accordance with DIN VDE 0110 Part 1. If higher pollution is expected, the devices must be installed in appropriate housings.

The user has to guarantee that the devices and the components belonging to them are mounted following these regulations. For operating the machines and installations, other national and international relevant regulations, concerning prevention of accidents and using technical working means, also have to be met.

The ABB Procontic devices are designed according to IEC 1131 Part 2. Meeting this regulation, they are classified in overvoltage category II which is in conformance with DIN VDE 0110 Part 2.

For the direct connection of ABB Procontic devices, which are powered with or coupled to AC line voltages of overvoltage category III, appropriate protection measures corresponding to overvoltage category II according to IEC-Report 664/1980 and DIN VDE 0110 Part 1 are to install.

Equivalent standards:

DIN VDE 0110 Part 1 ≙ IEC 664
DIN VDE 0113 Part 1 ≙ EN 60204 Part 1
DIN VDE 0660 Part 500 ≙ EN 60439-1 ≙ IEC 439-1

* VDE stands for "Association of German Electrical Engineers".

ABB Schalt- und Steuerungstechnik GmbH Heidelberg

Table of contents

1 Language of the ABB Procontic K200

1.1 Address assignment inputs/outputs (I/Os), flags, special functions, timers and counters

Designation	Number	Type	
Inputs Basic configuration	000–007 010–013	KR220	
	010–017	KR228	
	020–027	KR240	
	030–037 040–047	KR264	
Outputs Basic configuration	050–057	KR220	
	060–063	KR228	
	060–067	KR240	
	070–077	KR264	
Input expansions	100–107 110–117 120–127	EA240	
	130–137 140–147	EA264	①
Output expansions	150–157 160–167	EA240	
	170–177	EA264	
128 unbuffered flags	200–207 300–307 210–217 310–317 220–227 320–327 230–237 330–337 240–247 340–347 250–257 350–357 260–267 360–367 270–277 370–377		
248 buffered flags	400–407 600–607 410–417 610–617 420–427 620–627 430–437 630–637 440–447 640–647 450–457 650–657 460–467 660–667 470–477 670–677 500–507 700–707 510–517 710–717 520–527 720–727 530–537 730–737 540–547 740–747 550–557 750–757 560–567 760–767 570–577		②
8 special functions	770–777		
40 timers	T00–T07 T10–T17 T20–T27 T30–T37 T40–T47		
24 down counters	Z50–Z57 Z60–Z67 Z70–Z77		

Note: The maximum configuration of the system (80 I, 48 O) can only be achieved with use of the 07 KR 264 unit. If smaller basic configurations are used the number of input/output channels is reduced correspondingly.

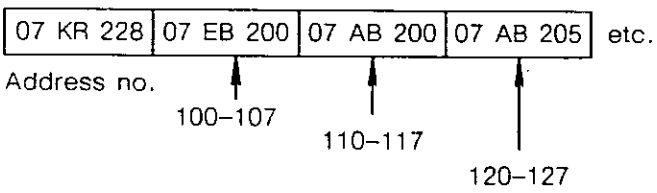
Example: Basic configuration 07 KR 228: 16 I, 12 O
Expansion module 07 EA 264: 40 I, 24 O
Total: 56 I, 36 O

Basic configuration 07 KR 220: 12 I, 8 O
and I/O modules, max. 40 I, 24 O
total max. 52 I, 32 O
May be realized with one basic configuration (07 KR 220) plus 5 EB XXX and 3 AB XXX (XXX = 200 or 205).

Note ①:

If instead of the expansion modules 07 EA 240 or 07 EA 264 I/O modules of the types 07 EB 200, 07 EB 205, 07 AB 200 or 07 AB 205 are used the address numbers of the I/O channels are determined by the components used.

Example:



The first module in addition to the basic configuration occupies the address range 100–107, the second module 110–117, etc. ragardless of whether it is an input or output module. The maximum configuration is

80 inputs/48 outputs. Output channels not occupied by hardware can be used in the program as additional flags.

Example: If the basic configuration 07 KR 228 is used the outputs 064 to 077 and 100 to 177 are available as unbuffered flags in addition.

Remark: Addressing of the ABB Procontic K200 follows an octal pattern. Only the numbers 0–7, 10–17, 20–27 are allowable.

Note ②:

For buffering a built-in gold capacitor is used. The buffer time is approx. 2 weeks.

1.2 Language structure (semantics)

The ABB Procontic K200 programming language is application oriented. It describes a control function sequence in a form easily understood by the user.

The programming language is matched specifically to control tasks.

The program consists of program records. Each record consists of program words (instructions).

Basic structure:

Word	(Instructions)	Smallest element of a program

Record
Word Word ... Word

A record consists of several words.

Word structure:

Word	
Operation part	Operand part

The operation part contains logic symbols and signs, and it is thus the processing instructions for the operands or for calling software functions.

The operand part contains variable identifiers and operand addresses, software function numbers and timer/counter values.

ABB Procontic K200 word		
Operation code	Operand code	Operand address

An ABB Procontic K200 word can consist of the following characters:

Logic symbol	Sign
&	N

Variable	Address (I/O or flag)
	01

Assign-ment symbol	Sign
=	N

Variable	Address (I/O or flag)
	205

Assign-ment symbol
=

Variable	No.	Value
T	05	035

Function
FUN

SW function number	Operand address
40	50

Record structure:

Record					
Interrogation			Assignment part		
!11	&N01	.../12	= 50	= 60	...= 52
Word 1	Word 2	...Wort n	Wort n+1	Word n+2	...Word n+m

A record must always consist of:
- start of record "!" (IF)
- at least one word in the interrogation
- at least one word in the assignment part (with assignment symbol "=" (THEN)) .

Smallest possible record:

Word 1	Word 2
! ——	= ——

A record can consist of any number of words – limited only by the available program memory space. Multiple assignments are possible.

1.3 Language elements (syntax)

1.3.1 Operation part

Logic symbol		Meaning
!	IF	Start of record
&	AND	AND function
/	OR	OR function
=	THEN	Assignment
STR	STORE	Store
FUN	FUNCTION	Assignment

Sign		Meaning
N	NOT	Negation

1.3.2 Operand part

Operand identifier	Meaning
T	Timer
Z	Counter

1.3.3 Operand address

00–047, 100–147	for inputs
50–077, 150–177	for outputs
T00 – T47	for timers T
T50 – T77	for counters Z
200 – 377	for unbuffered flags
400 – 767	for buffered flags
770 – 777	special functions
00 – 99	software function No.

1.3.4 Programming commands

No.	Command	Name	Function
1	!	if	always written at the
2	!N	if not	beginning of an instruction
3	STR	store	for buffering branches
4	STR N	store not	
5	&	and	Specifies that contacts
6	& N	and not	are to be connected in series (logical product)
7	/	or	Specifies that contacts
8	/N	or not	are to be connected in parallel (logical sum)
9	& STR	and store	Forms the logical sum of the branches connected in series
10	/ STR	or store	Forms the logical sum of branches connected in parallel
11	=	then	Result to output or flags
12	= N	then not	
13	FUN 0	loading constant	Load constant to register
14	FUN 00	Pulse generation (IMP)	For generating pulses out of steady signals; positive edge evaluation (e.g. setting of a memory)
15	FUN 02	set/reset (S/R)	Command for setting/resetting outputs and flags
16	FUN 03	R/S memory (R/S-Sp)	R/S memory for outputs and flags
17	FUN 04	MC set (MCS)	Commands for simplification
18	FUN 05	MC reset (MCR)	of complex logic operations
19	FUN 06	start of the branch instruction (SPR-A)	Branch instruction on the address starting with which the program shall not be executed
20	FUN 07	end of the branch instruction (SPR-E)	The end of the skipped program section is called up
21	FUN 7	≥ constant (≥ Konst) 1 bit (16 bits)	Register ≥ constant, then "1" signal to output/marker
22	FUN 8	= constant (= Konst) 1 bit (16 bits)	Contents of register = constant, then "1" signal to output/marker
23	FUN 9	< constant (< Konst) 1 bit (16 bits)	Contents of register < constant, then "1" signal to output/marker
24	FUN 11	Addition (ADD)	Add a number to the contents of the arithmetic register and take over the result into AR *) (16 bits)
25	FUN 12	Subtraction (SUB)	Subtract a number from the arithmetic register and take over the result into the AR (16 bits)
26	FUN 13	Multiplication (MULT)	Multiply the contents of the arithmetic register with a number and take over the result into the AR (16 bits)
27	FUN 14	Division (DIV)	Divide the contents of the arithmetic register with a number and take over the result into AR (16 bits)
28	FUN 22	Output word (=W)	Load word (value) from AR to outputs or markers (16 bits)
29	FUN 24	Bin-->BCD (BIN-BCD)	Convert status of the AR from Bin to BCD
30	FUN 25	BCD-->Bin (BCD-BIN)	Convert status of the AR form BCD to Bin
31	FUN 30	Load word (W LAD)	Load word from I/Os or flags into register (16 bits)
32	FUN 31	Load T/Z (T/Z LAD)	Load word (value) from T/Z into register (16 bits)
33	FUN 32	Output word (=Wc)	Load word from register to outputs or flags (16 bits)
34	FUN 33	Word to T/Z (=T/Z)	Load word from register to T/Z (16 bits)
35	FUN 34	Compare contents AR (W VGL)	If the value of the I/O's or markers is larger than or equal to the value of the AR, a FLAG is set, otherwise the FLAG remains cleared.

No.	Command	Name	Function
36	FUN 35	Compare contents AR (T/Z VGL)	If the value of the timers/counters is larger than or equal to the value of the AR, a flag is set, otherwise the flag remains cleared.
37	FUN 36	Load HZ (HZ Zähl)	Loads contents of high-speed counter into register (16 bits)
38	FUN 40	U/D counter (V/R-Z)	Up/down counter (4 decades) BCD coded
39	FUN 45	Dynamic memory (Dyn.SP)	Function for setting up memories with dynamic inputs (reset priority)
40	FUN 47	Shift register (Regist.)	Shift register (16 bits)
41	FUN 99	End	Identifies end of program
42	T/Z	Timer/counter	0-1 delay or downward counter

*) AR: Arithmetic register

Note: The functions FUN 0 to FUN 36 are called word-processing functions (16 bit instructions).

1.3.5 Special functions

No.	Number	Function
1	770	All outputs are turned OFF
2	771	Resetting of all buffered values (flags, timers, counters)
3	772	Cycle time oscillator (1 cycle impulse, 1 cycle pause)
4	773	0.1 oscillator
5	774	1 s oscillator
6	775	0.01 s oscillator
7	776	1 min. oscillator
8	777	One pulse after program start (cycle time)

1.3.6 Processing times of the various commands

No.	Designation	Time in µs	
		min.	max.
1	!, !N		3
2	&, &N		3
3	/, /N		3
4	STR, STR N		5
5	&STR		4.5
6	/STR		4.5
7	=		3
8	=N		5.5
9	=T/Z	22	56
10	=T/Z N	22	56.5
11	FUN 0		34.5
12	FUN 00	7.5	14.5
13	FUN 02	5	74
14	FUN 03	11	74
15	FUN 04		5
16	FUN 05		3
17	FUN 06		3.5
18	FUN 07		1.5
19	FUN 7		36.5
20	FUN 8		35
21	FUN 9		34
22	FUN 11		190.5
23	FUN 12		202.0
24	FUN 13		529.5
25	FUN 14		629.0
26	FUN 22		241.0
27	FUN 24		139.0
28	FUN 25		101.5
29	FUN 30	153	157
30	FUN 31		19
31	FUN 32	225	227
32	FUN 33	10.5	34.5
33	FUN 34	150.5	159.0
34	FUN 35	23	30
35	FUN 36		14.5
36	FUN 40	41.5	513
37	FUN 45	11	18
38	FUN 47	38.5	146.5
39	FUN 99	169.5	365
40	Transfer time for monitoring mode	250	940

1.4 Logic symbols

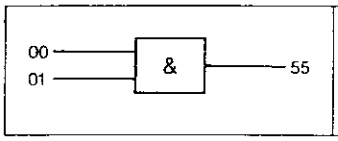
[!] IF, start of record

"IF" is always the first element of a record. It starts the interrogation part of the record.

& AND, conjunction

"AND" is specified between two variables and indicates that these are to be combined by an AND function.

Example:



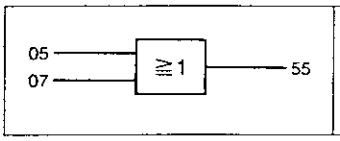
Program part	
!	00
&	01
=	55

Figure 1.1

[/] OR, disjunction

"OR" is specified between two variables and indicates that these are to be combined by an OR function.

Example:



Program part	
!	05
/	07
=	55

Figure 1.2

[=] THEN, assignment

"THEN" identifies the assignment part of a record. The result of the logical operations in the interrogation part of the record is assigned to the variable following this symbol. Multiple assignments are possible: each word in the assignment part receives the result of the interrogation part.

Example of a program:

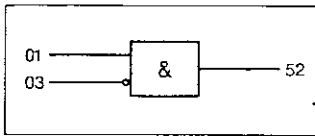
		Status
!	01	"1"
&	03	"1"
=	54	result of
=	53	interrogation part

1.5 Signs

[N] NOT, negation

"NOT" inverts the status of a interrogation or assignment.

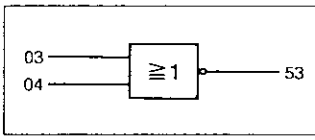
Example:



Program part	
!	01
&N	03
=	52

Figure 1.3

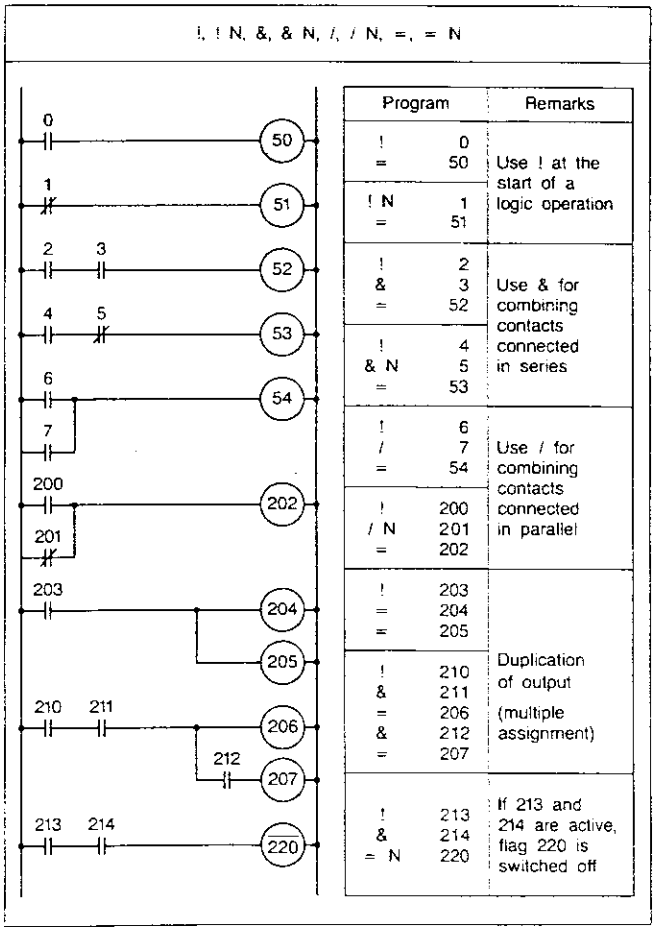
Example:



Program part	
!	03
/	04
=N	53

Figure 1.4

1.6 Summary examples for using the programming instructions !, !N, &, &N, /, /N, =, =N



Program		Remarks
! 0 = 50		Use ! at the start of a logic operation
! N 1 = 51		
! 2 & 3 = 52		Use & for combining contacts connected in series
! 4 & N 5 = 53		
! 6 / 7 = 54		Use / for combining contacts connected in parallel
! 200 / N 201 = 202		
! 203 = 204 = 205		Duplication of output (multiple assignment)
! 210 & 211 = 206 & 212 = 207		
! 213 & 214 = N 220		If 213 and 214 are active, flag 220 is switched off

Figure 1.5 (a)

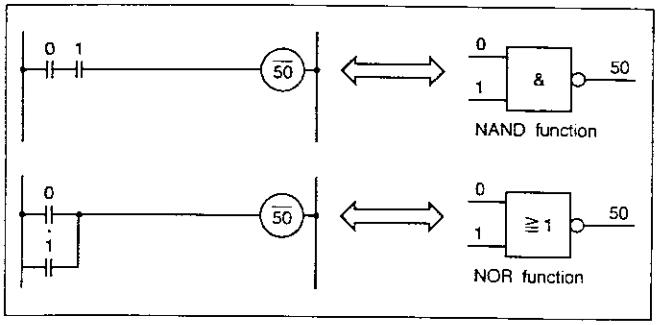


Fig. 1.5

1.7 Logical link of parallel and serial blocks with STR or STRN

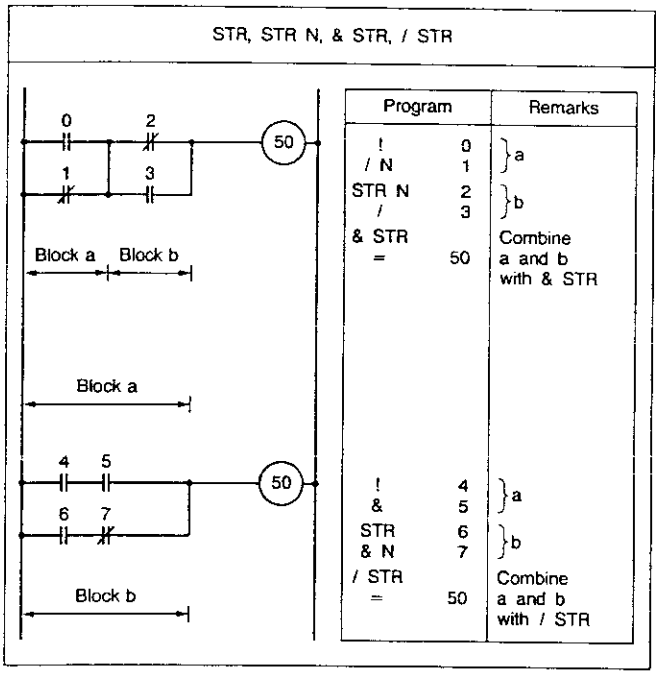


Figure 1.6

a) After the circuit has been subdivided into blocks, the first block (in the above examples always block a) is used for logic operations as usual. At the beginning of the following blocks (in the above examples block b follows) STR or STRN is written in each case.

   How the blocks are to be logically linked has not yet been decided with this input. Only prior to output assignment one determines by input of &STR and/or /STR whether a & and/or / logic combination of the blocks is intended, i. e. in case of series connection one closes with &STR and in case of parallel connection with /STR.

b) &STR or /STR may be written seven times in succession.

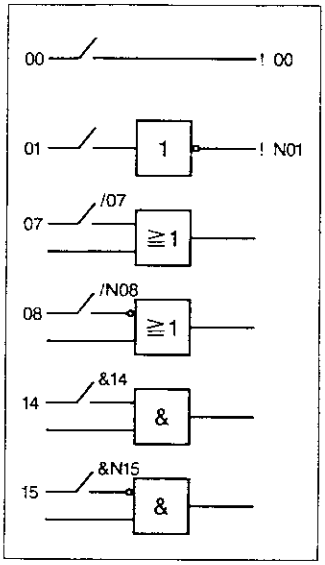
c) A syntax error will appear if after a STR or STRN in the program no &STR or /STR follows. This does not apply to to counters, times, shift registers, memories FUN 03 and FUN 45.

1.8 Inputs

Inputs

Address range: 000 ... 047
100 ... 147

Possible combinations of logical symbols and signs:



! ...

!N ...

/ ...

/N ...

& ...

&N ...

Figure 1.7: Representation of input combinations by binary functions

Caution: A program word containing the address of an input channel causes the status of this input ("0" or "1") to be interrogated.

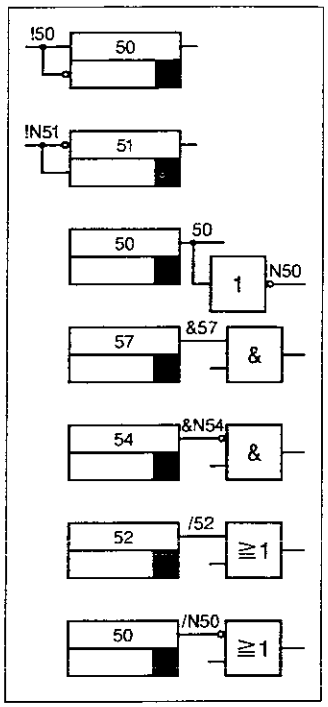
The input variable may not be used in the assignment part of a record (inputs can only be processed, but not directly controlled by the program).

1.9 Outputs

Outputs

Address range: 050 ... 077
150 ... 177

Possible combinations of logical symbols and signs:



! ...

!N ...

= ...
=N ...

& ...

&N ...

/ ...

/N ...

Figure 1.8: Representation of output logic functions by binary functions

Caution: The output variable can be used both in the condition part and in the assignment part of a record.

In the interrogation part, a program word containing the output variable causes the output channel identified by the address number to be interrogated for its status ("0" or "1").

Two forms are possible in the assignment part:

a. Non buffered (reference = ... , =N...)
The result of the interrogation part is assigned to the output channel identified by the address number. An non buffered output may be used only once in a program.

b. Buffered output (reference, for example, using FUN 02, FUN 03, = ...)
If the set condition is fulfilled, the output channel is set to "1" and is reset again only if the reset condition is fulfilled.

Caution: An output may be assigned only once in a program.

1.10 Variable T – timer

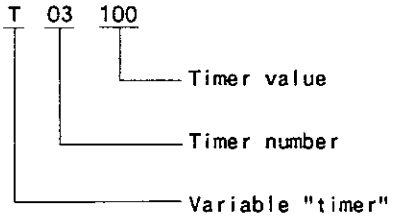
T Timer

Address range	value range
T00 ... T47	0.01 ... 9.99 seconds
	0.1 ... 99.9 seconds
	1 ... 999 seconds

The timing elements (with ON delay) are programmed with the variable T and a 2-decade number (T00-T47), i. e. 40 timing elements can be defined, since addressing in the PROCONTIC K200 is effected in octal mode.

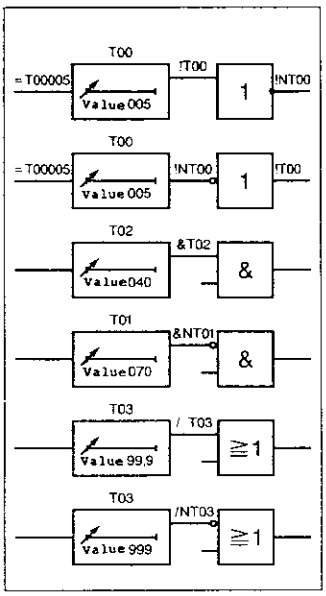
For programming the 3-decade time a decimal point is used if necessary.

Example: T 03 100



In the above example the timing element no. 03 has been programmed with 100 s.

Possible combinations of logic symbols and signs:



! T...
= T...
=N T...

!N T...

& T...

&N T...

/ T...

/N T...

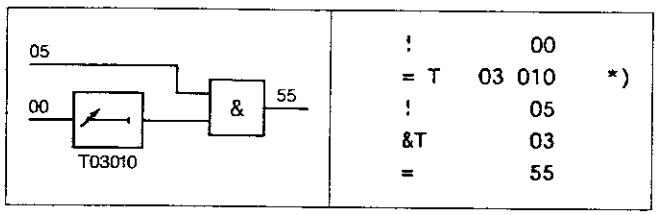
Figure 1.9: Representation of timer combinations by binary functions

Caution: The timers are set up as "0-1" delay circuits and the time values are set by software.

The reset time of a timer function is shorter than the processing time for one word. The preset time elapses if the

assignment time contains "T" and the activation condition exists longer than the delay time. Interrogation of "T" results in the value "0" as long as the activation condition for "T" is not fulfilled or the time is running.

Interrogation of "T" results in the value "1" if the time has elapsed and the activation condition is still fulfilled. If the activation condition is interrupted during the delay time, the timer is restarted when the activation condition is again fulfilled.



Programm

```
!        00
= T   03 010    *)
!        05
&T       03
=        55
```

Figure 1.10

*) timer 03
with 10 s

Multiple use of the timers is possible if

a. the same time value is required more than once in a program
b. the individual activation conditions are combined by OR function to a single assignment
c. each time interrogation is combined by an AND function with related activation conditions
d. simultaneous activation by several conditions is impossible

Example:

```
!        00
/N       05
= T   01 050          assignment
!        00
&        T01          interrogation
=        52

!N       05
&        T01          interrogation
=        53
```

Caution: Multiple assignment of timer functions is not permitted.

The applications of timers can be extended by:
- chaining together of times to increase the value (see software modules)
- use as "1-0" delays, blockers, etc. (see software modules)

1.11 Variable Z – counter

Z Counter

Address range	Counter value range
Z50 ... Z77	001 ... 999
Z70 ... Z77	001 ... 9999

The down counters are programmed with the variable Z and a 2-decade number (Z50 to Z77), i. e. 24 counting elements may be defined, since the PROCONTIC K200 is addressed in octal mode. The 3-decade count value may be programmed in the counting range between 0 and 999 as the maximum.

Example: Z 55 672

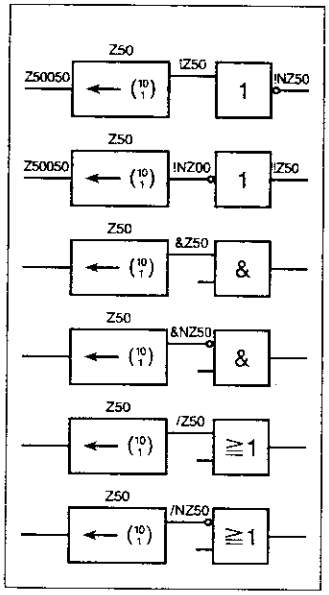
Counter value

Counter No.

Variable "counter"

In the above example counter no. 55 has been programmed with the initial counter reading 672.

Possible combinations of logic symbols and signs:



!	Z...
=	Z...
=N	Z...
!N	Z...
&	Z...
&N	Z...
/	Z...
/N	Z...

Figure 1.11: Representation of counter combinations by binary functions

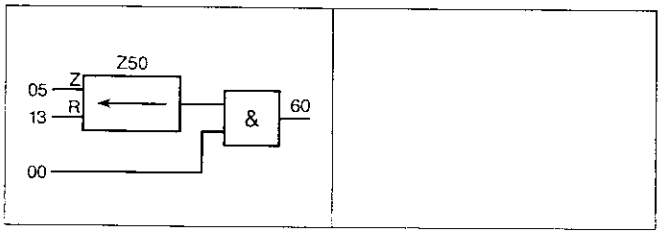


Figure 1.12

Caution: The counters are designed as downward counters and the count values are entered in the program when the counters are assigned. When a counter is interrogated, the output or flag becomes "1" if the counter value reaches zero.

The counter value is stored in the case of a voltage failure. The counter can be reset and set to its initial value with the command STR...

Example:

```
!          05          count input
STR        13          reset input
= Z    50 051          counter 50, value 51)
!         Z50          check whether Z50 is
&          00          zero
=          60          output 60
```

Calling the four-decade down counters:
The counters Z70 – Z77 can be used as 4 decade down counters. This is done as follows:

During programming, omit the tenth digit (7) and enter a decimal point after the counter number (0–7).

Example: Counter 74 is to be programmed with the count value 1350

Solution: = Z 4.1350

If, in contrast, counter 74 is to be programmed with a three-decade number such as 256, this is done as follows:

Solution: =Z74256

To interrogate the counter output, the counter number 70 to 77 is used in both cases.

Example: Counter 75 is to be programmed with 3560. When it reaches the value zero, output 50 is to be set.

Program:

```
!          773          count input
STR          3          reset input
=Z     5.3560           counter 75, value 3560
!Z          75          if counter 75 = zero
=           50          set output 50
```

1.12 Summary example for T/Z times/counters

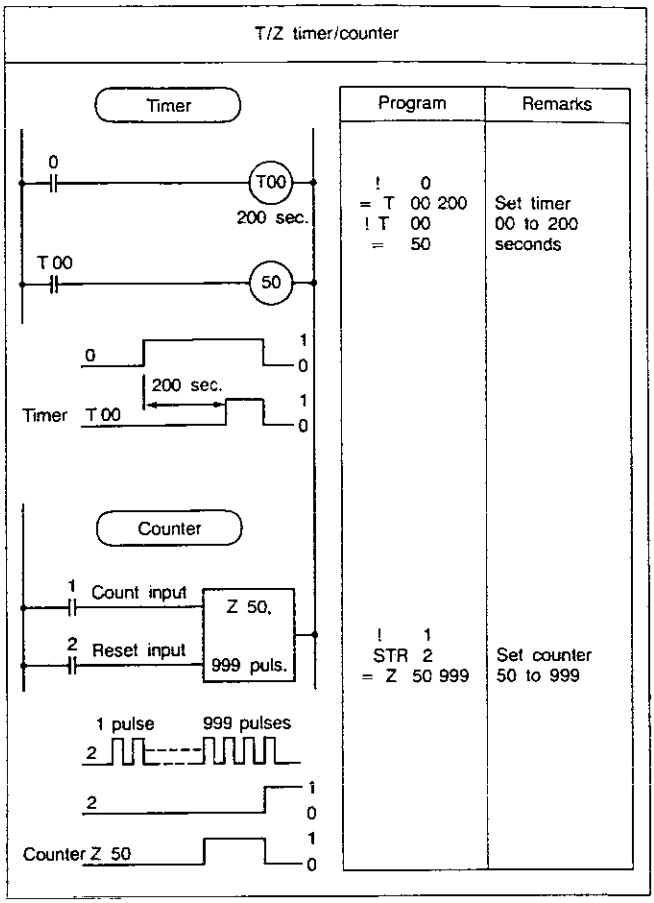


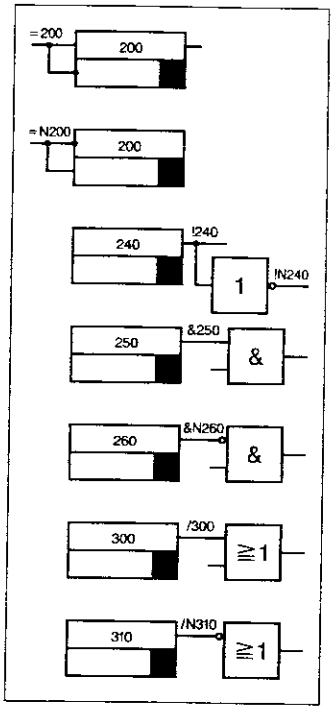
Figure 1.13

1.13 Unbuffered flags, 128 flags

Unbuffered flags

Address range: 200 ... 377

Possible combinations of logic symbols and signs:



= ...

=N ...

! ...

!N ...

& ...

&N ...

/ ...

/N ...

Figure 1.14: Representation of flag combinations by binary functions

Flags are auxiliary functions which are used, either buffered or unbuffered, for combining parts of a program or for storing a status (for example with FUN 02, FUN 03).

Example:

```
!        00
&        03
&        T07
=        250
```

In the assignment part, flag 250 is set to "1" if the conditions in the interrogation part are fulfilled or to "0" if they are not fulfilled.

Two different forms of use are possible in the assignment part:

a. Non buffered (reference = 200, =N250)

The value of the interrogation part is assigned to the flag identified by the address.

Multiple assignment of an unbuffered flag in a program is not permitted. Any necessary multiple assignments are to be grouped together in an OR function.

Example:

```
!        05
/        02
/N       20
=        200
```

b. Buffered (reference FUN 02, FUN 03)
When the setting condition is fulfilled the flag is set to "1"-signal and reset only when the reset condition has been fulfilled. Multiple assignment of buffered flags in a program is not permitted.

Examples:

```
!        00
&        T02
FUN      02
=        200

!        04
&        06
FUN      02
=N       200
```

If the conditions in the interrogation part are fulfilled, FUN 02 in the assignment part sets flag 200 to "1" (1st record) or resets it to "0" (2nd record).

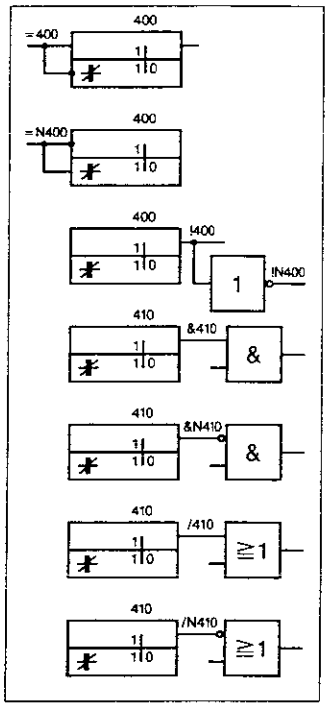
Caution: Only <u>one</u> of the above mentioned two possible forms of use may be utilized for a given flag (such as 205) in a program.

1.14 Buffered flags, 248 flags

Buffered flags

Address range: 400 ... 767

Possible combinations of logic symbols and signs:



= ...

=N ...

! ...

!N ...

& ...

&N ...

/ ...

/N ...

Figure 1.15: Representation of buffered flag combinations by binary functions

Flags are auxiliary functions which are used with or without memory (e.g. FUN 02, FUN 03) for combination of parts of a program or for storage of status. Buffered flags retain their information even in case of power failure or the supply voltage is switched off.

Example:

```
!        00      set input
STR      02      reset input
FUN      03
=       400
!       400
=        50
```

This example shows a dominant reset memory which is buffered. If input 00 is active, set flag 400. Then set output 50. If input 02 is active, reset 400 and 50.

Two forms of use are permitted in the assignment part:

a. Non buffered (reference = 400, =N400)

The buffered flag identified by the address receives the result of the interrogation part.

A non buffered flag can be assigned only once in a program. However, it can be interrogated as often as required. Any required multiple assignments are to be combined with an OR function.

Example:

```
!        05
/       200
/N       50
=       400
```

b. Buffered (reference with FUN 02, FUN 03)

If the set condition is fulfilled, the flag is set to "1" and is reset only when the reset condition is fulfilled.
Buffered flags with memory may be assigned only once in a program.

Example:

```
!        00      set input
STR      02      reset input
FUN      03      memory function with dominant
                 reset
=       400      flag 400
```

Caution: A buffered flag (such as 405) may be used in only one of the two forms in a program.

1.15 FUN 02 (set/reset) and FUN 03 (RS memory)

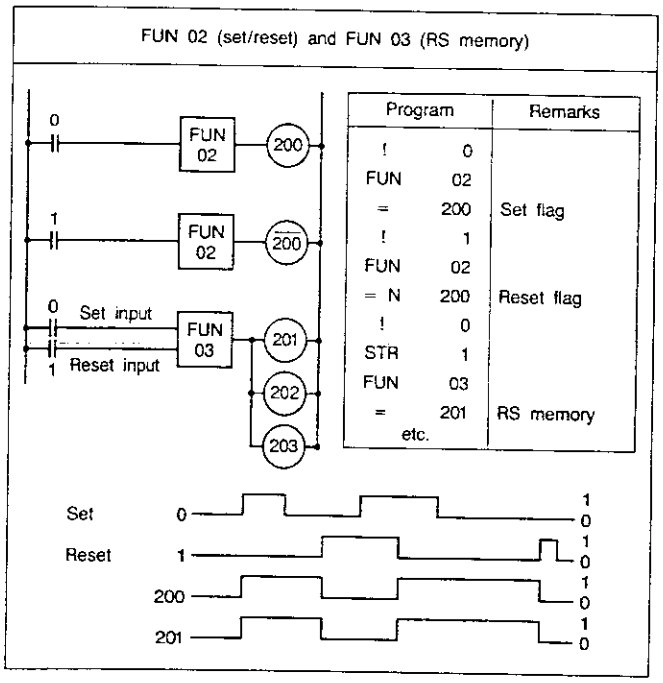


Figure 1.16

a. FUN 02 is used for setting or resetting an output or a flag. This is explained with the aid of a flow chart below.

aa. If a "1" signal is present at input 0, output 200 is set.

ab. If a "0" signal is present at input 1, nothing happens; the output remains set.

ac. If "0" signal is present at input 0, nothing happens.

ad. If "1" signal is present at input 1, output 200 is reset.

b. FUN 03 is an RS memory.

Caution! The outputs of the above functions can be duplicated. FUN 02 and FUN 03 have reset priority.

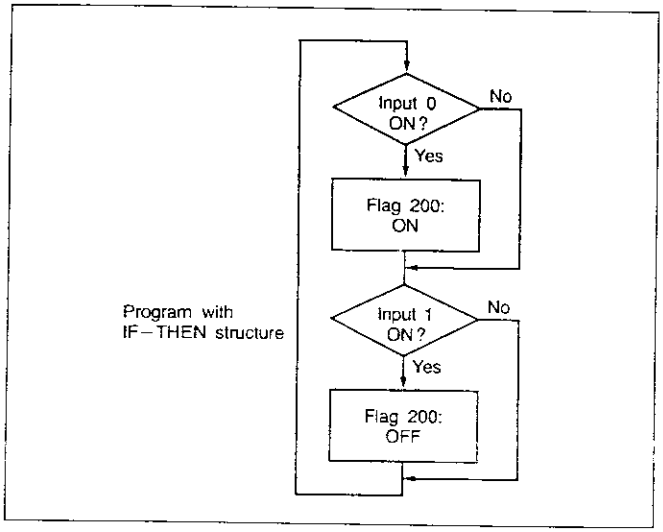


Figure 1.17

1.16 FUN 04 (MC set) and FUN 05 (MC reset)

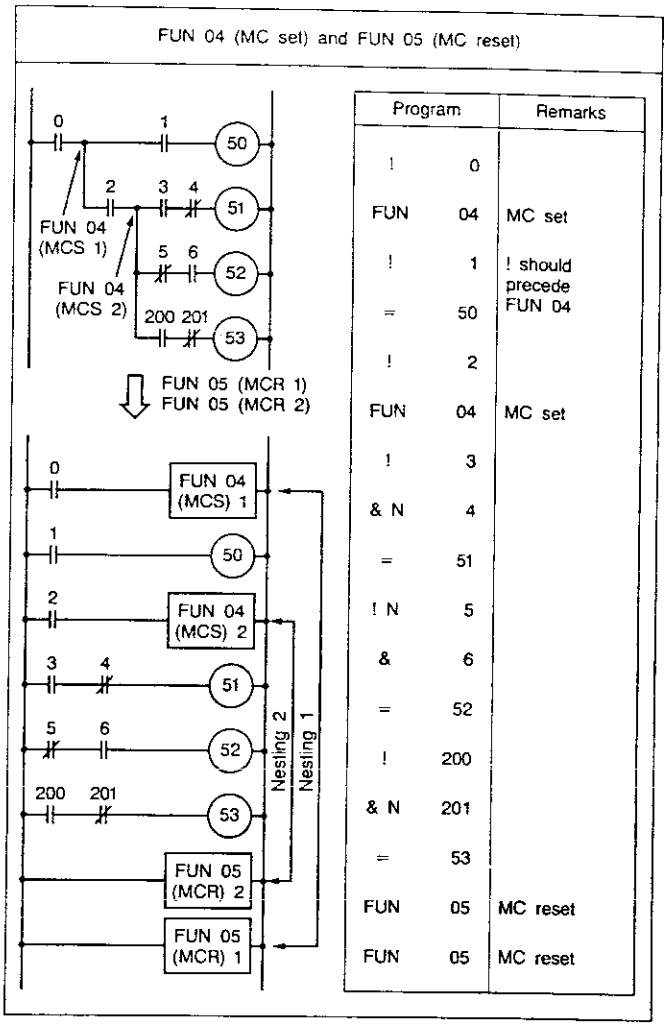


Figure 1.18

a. FUN 04 and FUN 05 always are to be used in pairs, as the system will otherwise respond with a syntax error.

b. The maximum nesting is 3. If nesting is more than 3, a syntax error occurs.

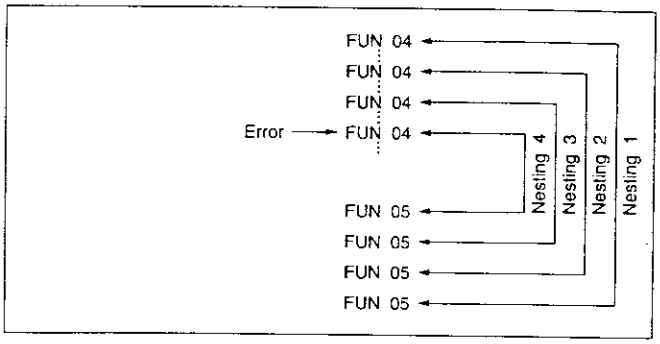


Figure 1.19

1.17 Jump commands FUN 06 and FUN 07

FUN 06 Jump to address

FUN 06 is specified in the assignment part of a record. If the conditions of the record are fulfilled, all program words between FUN 06 and FUN 07 are skipped. If the conditions of the record are not fulfilled, the program words between FUN 06 and FUN 07 are processed just like a normal part of the program.

Caution: All variables between FUN 06 and FUN 07 remain unchanged if the conditions for FUN 06 are fulfilled; this means that any variables which are set in this part of the program will remain set even if the interrogation condition is no more given.

FUN 07 End of jump

FUN 07 is specified alone, without further logic operations or assignments. A jump command initiated with FUN 06 must be terminated with FUN 07.

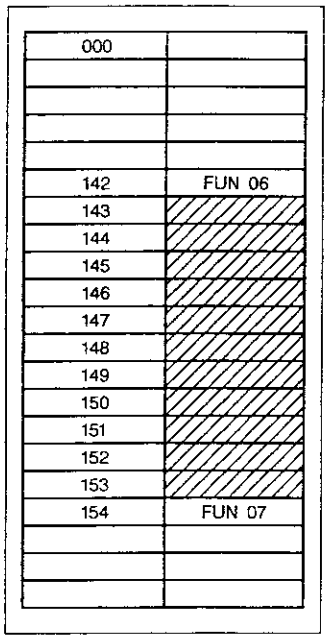


Figure 1.20

Example:

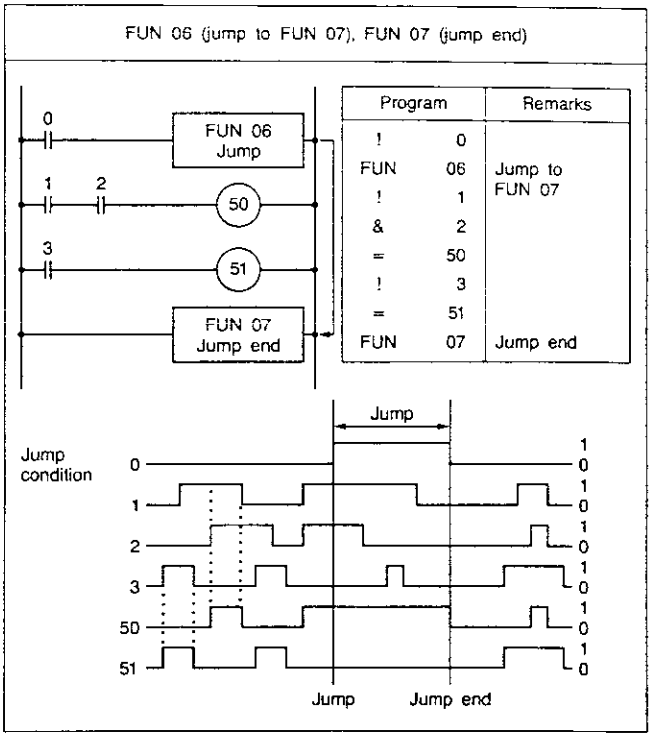


Figure 1.21

a. FUN 06 and FUN 07 must always be used together in pairs. A syntax error will otherwise result. Nesting should be programmed as shown in the figure.

b. If the jump command is active, the part of the program between FUN 06 and FUN 07 is not executed and all outputs in this part of the program remain unchanged.

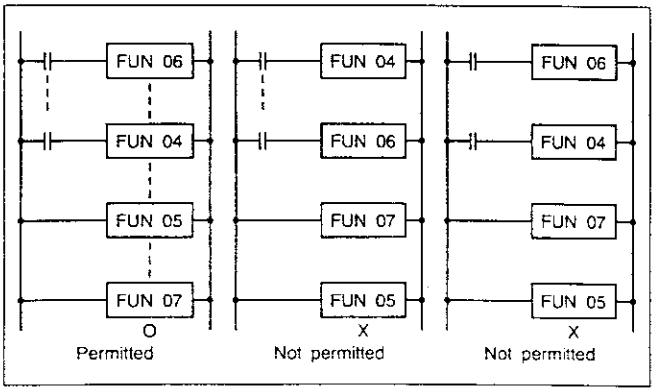


Figure 1.22

1.18 FUN 00 (Pulse Generation) and FUN 99 (End of Program)

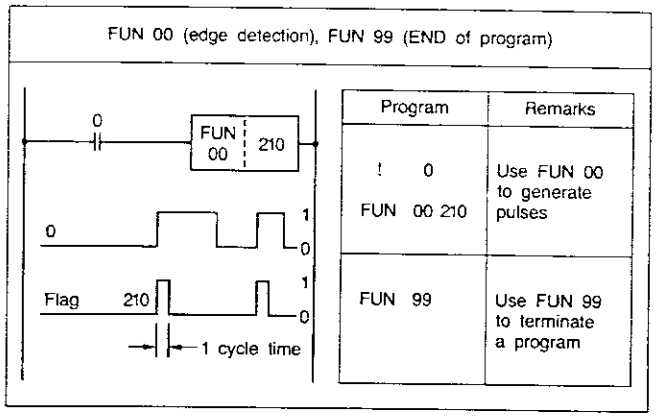


Figure 1.23

a. If pulses have to be generated from continous signals (e. g. to set a memory), this is done with FUN 00.

b. FUN 99 is used to indicate the end of the program. This function is normally not necessary.

FUN 99 is used to identify the end of the user program. It does not have to be programmed. However, FUN 99 can be inserted anywhere in the user program, for example during commissioning, for testing of program sections. This accelerates execution of the program, as the part of the user program following FUN 99 is not executed.

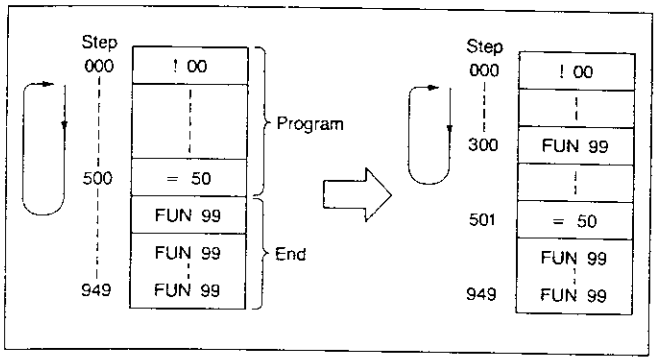


Figure 1.24

Notes

2.1 AND function

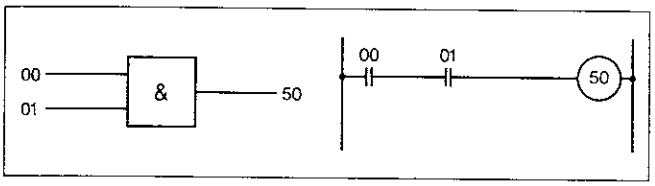


Figure 2.1

!	00	Start of record	Input 00 = "1"?
&	01	AND	Input 01 = "1"?
=	50	Assignment	Output 50 = "1"

In this example, inputs 00 and 01 are checked for the signal "1" and linked conjunctively. Together the result of this interrogation is assigned to output 50.

00	01	Output 50
0	0	0
0	1	0
1	0	0
1	1	1

2.2 OR function

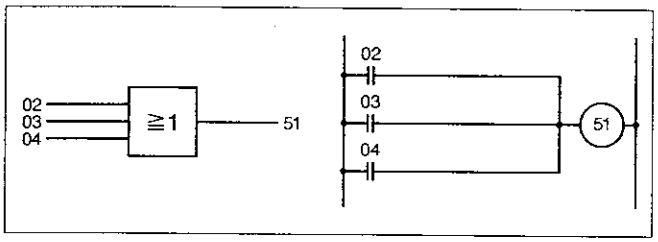


Figure 2.2

!	02	Start of record	Input 02 = "1"?
/	03	OR	Input 03 = "1"?
/	04	OR	Input 04 = "1"?
=	51	Assignment	Output 51 = "1"

In this example, inputs 02 to 04 are tested for signal "1" and linked disjunctively. Together the result of this interrogation is assigned to output 51.

02	03	04	Output 51
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2.3 Combination of logic functions

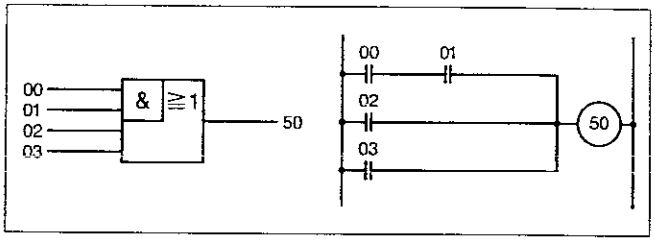


Figure 2.3

!	00	Start of record	Input 00 = "1"?
&	01	AND	Input 01 = "1"?
/	02	OR	Input 02 = "1"?
/	03	OR	Input 03 = "1"?
=	50	Assignment	Output 50 = "1"

00	01	02	03	Output 50
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Caution: If several contacts are connected in series in the parallel branches (e. g. 02), they must be combined with STR commands.

2.4 NOT function

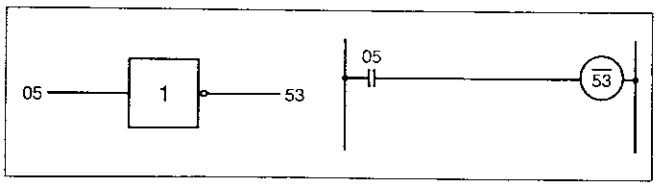


Figure 2.4

!	05	Start of record	Input 05 = "1"?
=N	53	Assignment	Output 53 = "0"

05	Output 53
1	0
0	1

All previously described logic functions generate the status "1" if the inputs are "1" and the status is "0" if the inputs are "0". These functions are called "normally open" contacts in a relay circuit.

However, all modern controls require functions which deliver the status "1" for "0" at the inputs or the status "0" for "1" signals at the inputs. These functions are equivalent to the "normally closed" contacts in a relay circuit. In the programming language, the character "N" provides this function.

Instead of	!	05
	=N	53
it is possible to program, with the same result (De Morgan):	!N	05
	=	53

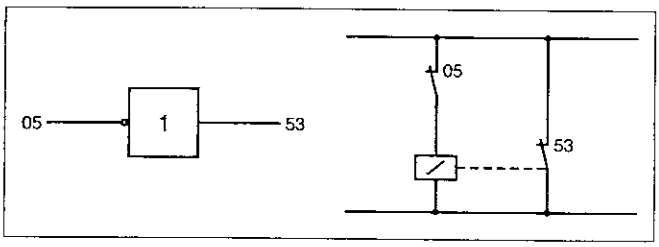


Figure 2.5

2.5 NAND function

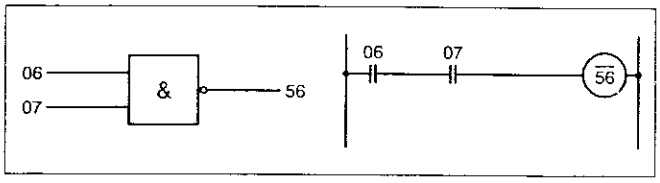


Figure 2.6

!	06	Start of record	Input 06 = "1"?
&	07	AND	Input 07 = "1"?
=N	56	Assignment	Output 56 = "0"

In this example, inputs 06 and 07 are checked for "1" signal and combined with the AND function. The result of this operation is negated and assigned to output 56.

06	07	Output 56
0	0	1
0	1	1
1	0	1
1	1	0

Instead of	!	06
	&	07
	=N	56
it is possible to program, with the same result (De Morgan):	!N	06
	/N	07
	=	56

2.6 NOR function

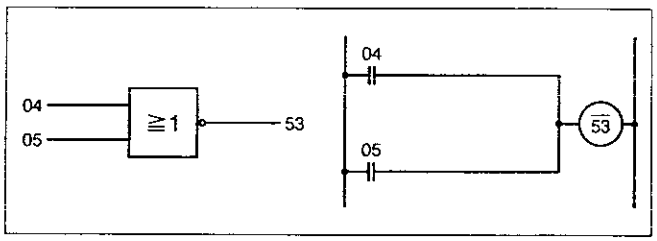


Figure 2.7

!	04	Start of record	Input 04 = "1"?
/	05	OR	Input 05 = "1"?
=N	53	Assignment	Output 53 = "0"

In this example, the inputs 04 and 05 are checked for a "1" signal and combined with the OR function. The result of this operation is negated and assigned to output 53.

04	05	Output 53
0	0	1
0	1	0
1	0	0
1	1	0

Instead of	!	04
	/	05
	=N	53
it is possible to program, with the same result (De Morgan):	!N	04
	&N	05
	=	53

2.7 AND before OR function

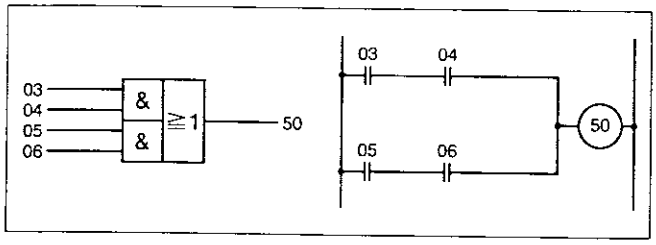


Figure 2.8

!	03	Start of record	Input 03 = "1"?
&	04	AND	Input 04 = "1"?
STR	05	OR	Input 05 = "1"?
&	06	AND	Input 06 = "1"?
/	STR	/STR	OR and store
=	50	Assignment	Output 50 = "1"

This AND before OR function is initiated with the STR command (and buffered). The /STR before the assignment executes an OR function.

03	04	05	06	Ausgang 50
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	1
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	1
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

2.8 OR before AND function

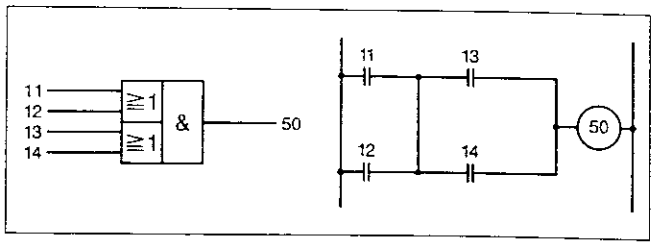


Figure 2.9

!	11	Start of record	Input 11 = "1"?
/	12	OR	Input 12 = "1"?
STR	13	STR	Input 13 = "1"?
/	14	OR	Input 14 = "1"?
&	STR	& STR	AND and store
=	50	Assignment	Output 50 = "1"

The OR before AND function is initiated with the STR command (and buffered). The & STR before the output assignment executes an AND function.

11	12	13	14	Ausgang 50
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	1
0	1	1	0	1
1	1	1	0	1
0	0	0	1	0
1	0	0	1	1
0	1	0	1	1
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

2.9 Exclusive OR circuit

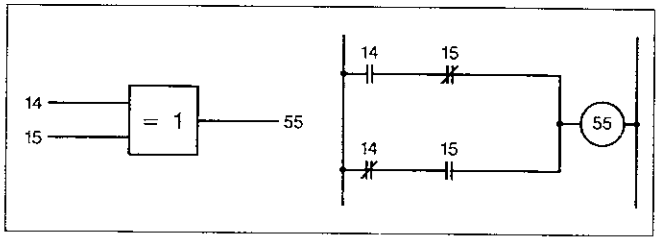


Figure 2.10

!	14	Start of record	Input 14 = "1"?
&N	15	AND	Input 15 = "0"?
STRN	14	OR	Input 14 = "0"?
&	15	AND	Input 15 = "1"?
/	STR	/STR	OR and store
=	55	Assignment	Output 55 = "1"

Output 55 is set to "1" if input 14 is "1" and input 15 is "0" or if input 14 is "0" and input 15 is "1".

14	15	Output 55
0	0	0
1	0	1
0	1	1
1	1	0

2.10 Equivalence circuit

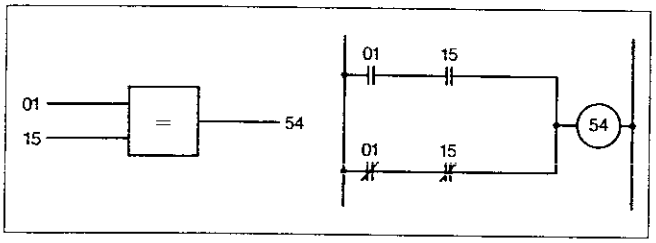


Figure 2.11

!	01	Start of record	Input 01 = "1"?
&	15	AND	Input 15 = "1"?
STRN	01	OR	Input 01 = "0"?
&N	15	AND	Input 15 = "0"?
/	STR	/STR	OR and store
=	54	Assignment	Output 54 = "1"

Output 54 is set to "1" if input 01 is "1" and input 15 is "1" or if input 01 is "0" and input 15 is "0".

01	15	Output 54
0	0	1
1	0	0
0	1	0
1	1	1

2.11 Driving of several outputs (output duplication)

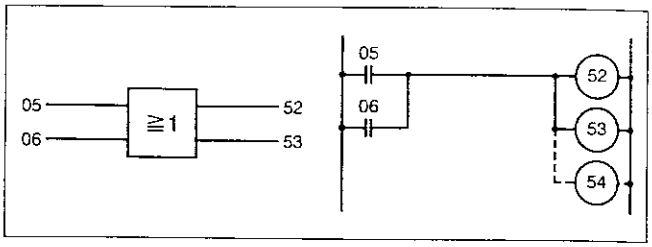


Figure 2.12

!	05	Start of record	Input 05 = "1"?
/	06	OR	Input 06 = "1"?
=	52	Assignment 1	Output 52 = "1"?
=	53	Assignment 2	Output 53 = "1"

Any number of assignments can be specified, with or without negation. The assignments always refer to the result of the condition part of the record before the first assignment symbol.

		Outputs	
05	06	52	53
0	0	0	0
1	0	1	1
0	1	1	1
1	1	1	1

3.1 Memories for outputs or flags – dominant reset

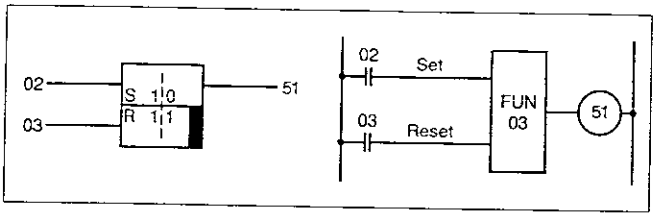


Figure 3.1

Program:

```
!       02
STR     03
FUN     03
=       51
```

If input 02 is "1" and input 03 is "0", output 51 is set by FUN 03. Output 51 remains set to "1", when input 02 carries "0" signal again.

The output memory is reset if input 03 is "1".

If both inputs (02 and 03) are "1", the memory is reset (dominant reset).

02	03	Output 51
0	0	unchanged
0	1	0
1	0	1
1	1	0

If flags with address numbers from 400 to 767 are used, then the memories are buffered, e. g. flags retain their information even in case of power failure or the supply voltage is switched off.

3.2 Memories for outputs or flags – dominant set

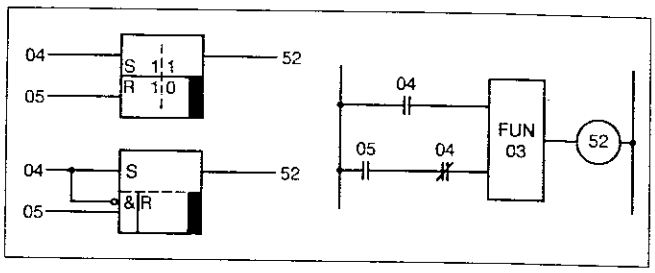


Figure 3.2

Program:

```
!       04
STR     05
&N      04
FUN     03
=       52
```

If input 04 is "1", output 52 is set and stored with FUN 03. When input 05 carries "0" signal, output 52 remains set to "1" even if input 04 is reset to "0" signal.

If input 05 is "1" and input 04 is "0", output 52 is reset.

If both inputs (04 and 05) are "1", the memory is set (dominant set).

05	04	output 52
0	0	unchanged
0	1	1
1	0	0
1	1	1

If flags with address numbers from 400 to 767 are used, then the memories are buffered.

3.3 FUN 45 – Memory with dynamic input

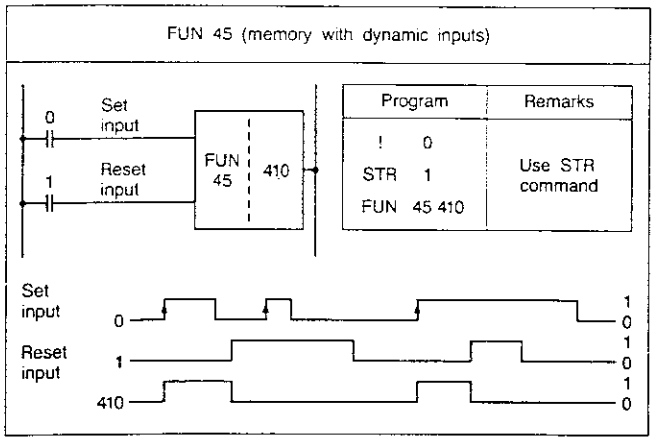


Figure 3.3

a. FUN 45 is a function for generating memories with dynamic inputs (memories with flags 200 – 377, buffered flags with 400 – 767, in each case with dynamic inputs). The function FUN 45 can also be sent directly to the output (not protected from power failure).
Example: FUN 45 50

b. The difference between this and a normal self-hold circuit is shown with the following diagram.

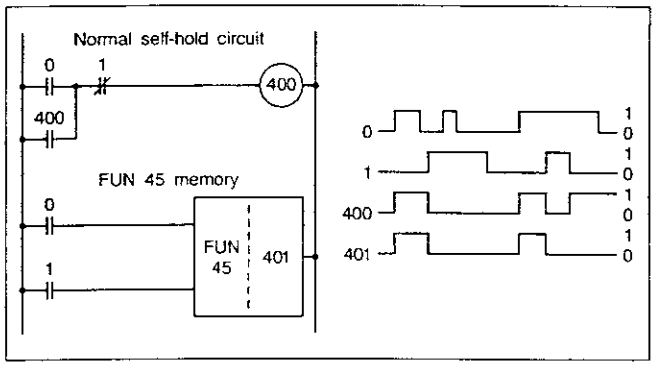


Figure 3.4

4.1 Edge evaluator, positive going edge (0–1)

Program:

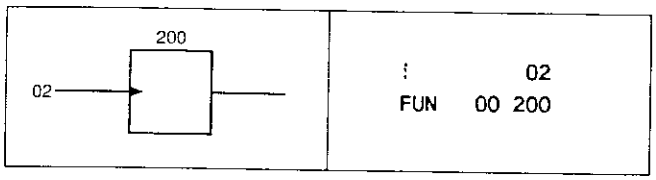


Figure 4.1

When the "0–1" edge at input 02 is detected, flag 200 is set for the duration of one program cycle.

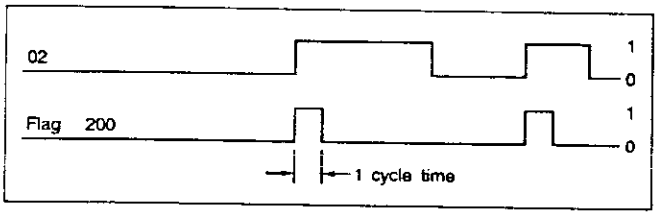


Figure 4.2

4.2 Edge evaluator, negative going edge (1–0)

Program:

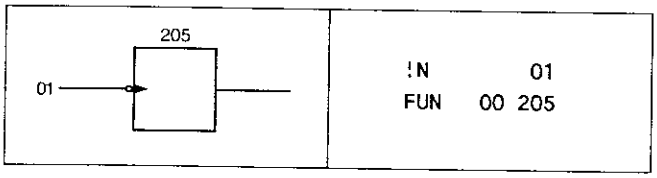


Figure 4.3

When the "1–0" edge at input 01 is detected, flag 205 is set for the duration of one program cycle.

4.3 Latching relay

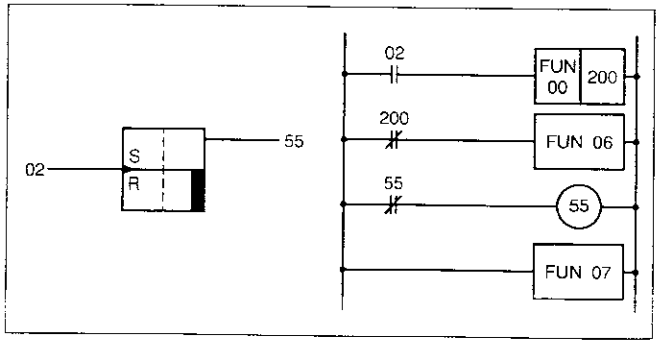


Figure 4.4

Program:

```
!          02
FUN   00 200
!N         200
FUN        06
!N         55
=          55
FUN        07
```

If there is a change from "0" signal to "1" signal at input 02, function FUN 00 generates, from this continous signal, a pulse with a width of the cycle time. Output 55 is set. When the next pulse arrives, the output is reset, and so on. In all cases, the "0–1" transition of the input signal initiates the change of the output signal.

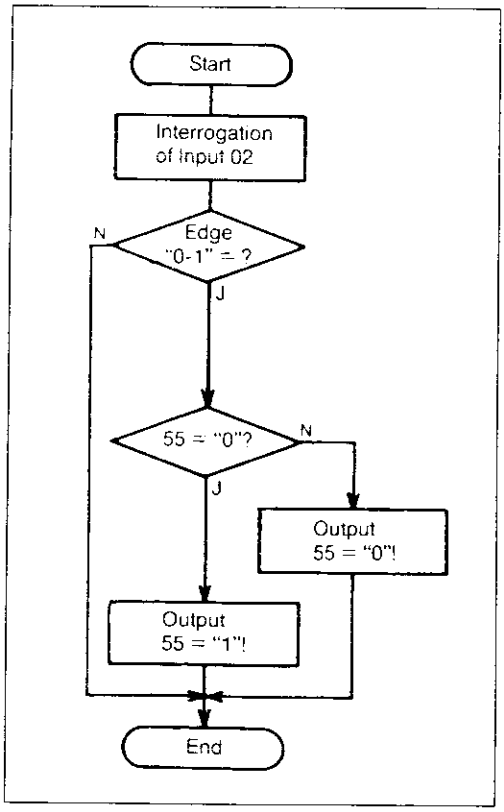


Figure 4.5

Notes

5.1 On delay (0–1)

Program:

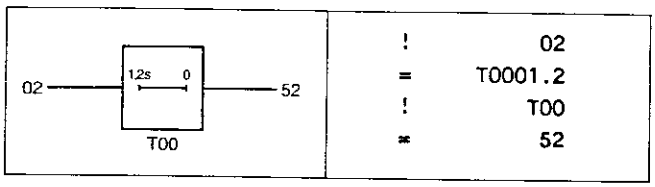


Figure 5.1

In the case of an on delay, the time starts when input 02 is "1". Output 52 is set when the delay has elapsed and if the input signal is still present. This is done by interrogating the elapsed time.

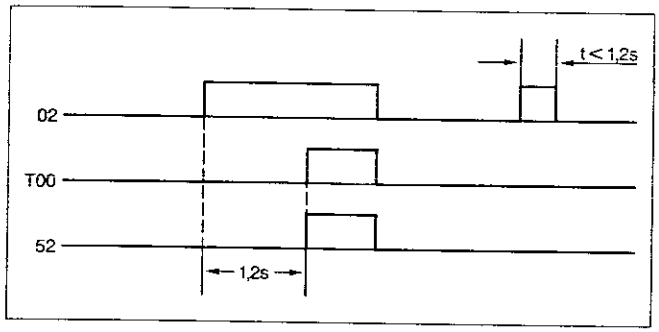


Figure 5.2

If the duration of the input signal is shorter than the delay, output 52 remains off.

5.2 Off delay (1–0)

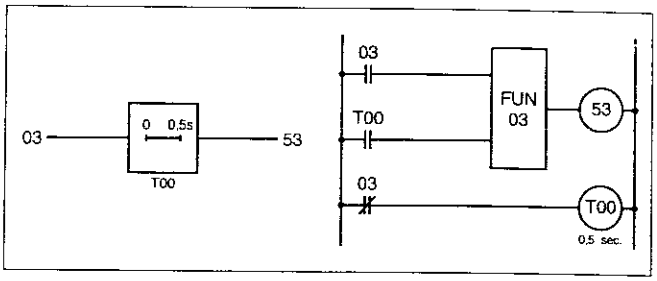


Figure 5.3

Program:

```
!       03
STR     T00
FUN     03
=       53
!N      03
=       T0000.5
```

In the case of the off delay, the time starts when input 03 is "0". Output 53 is set immediately if there is a "1" signal at input 03. Output 53 is reset after the time delay has elapsed.

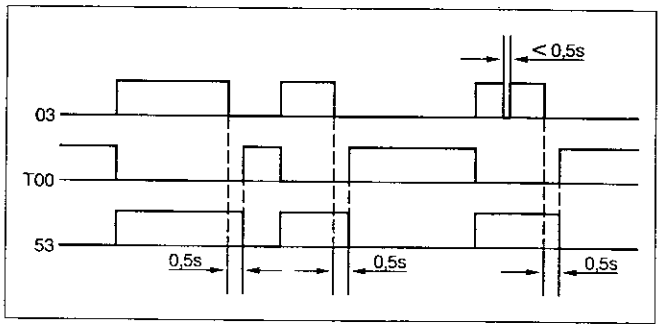


Figure 5.4

5.3 On/off delay with 1 timer

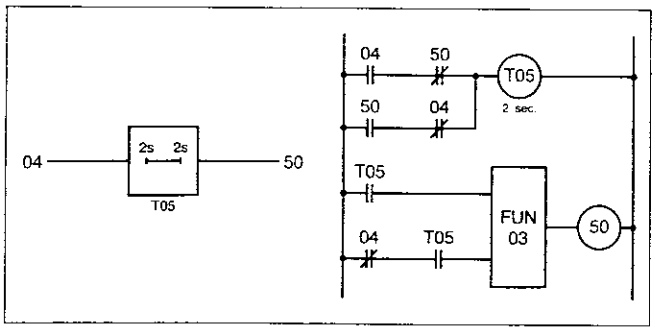


Figure 5.5

Program:

```
!          04
&N         50
STR        50
&N         04
/         STR
=      T05 002
!          T05
STRN       04
&          T05
FUN        03
=          50
```

When a "1" signal is present at input 04, the turn-on delay time T05 starts. When the time has elapsed output 50 is set.

When the "1" signal at input 04 becomes a "0" signal, the same programmed time runs once again (turn-off delay time) and output 50 is reset.

When the input signal is present for less than (<) the programmed time, the output retains its previous status.

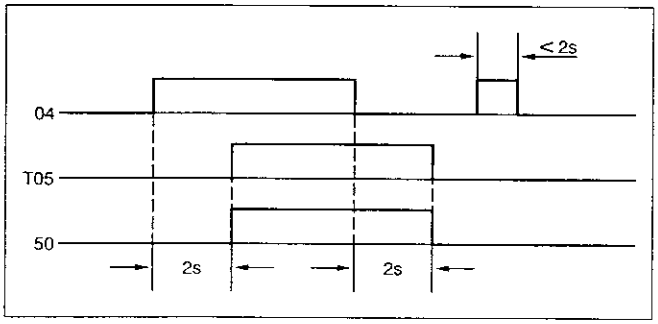


Figure 5.6

5.4 On/off delay with 2 timers

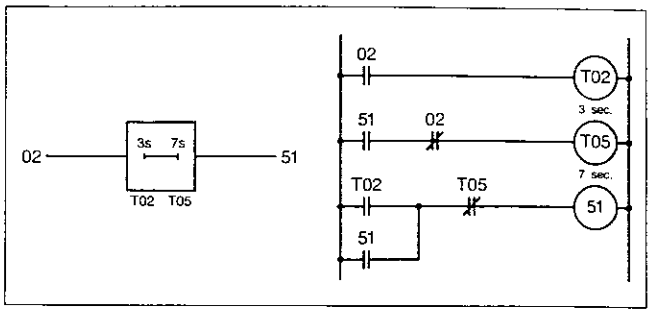


Figure 5.7

Program:

```
!          02
=      T02 003
!          51
&N         02
=      T05 007
!          T02
/          51
&N         T05
=          51
```

With a "1" signal at input 02 the turn-on delay time T02 starts. When this time has elapsed, output 51 is set. When the "1" signal at input 02 turns into a "0" signal, the turn-off delay time T05 will start. When the turn-off delay time T05 has elapsed, output 51 is reset.

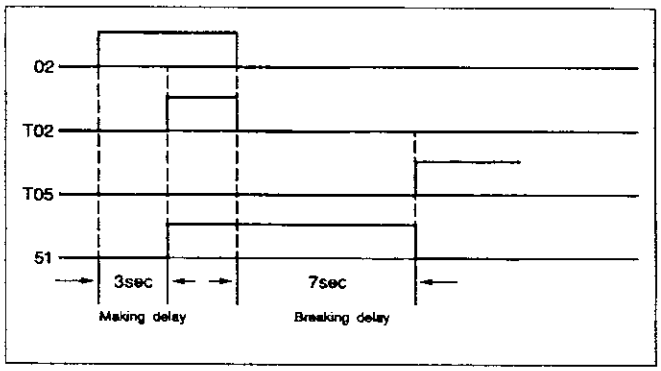


Figure 5.8

5.5 Blocker

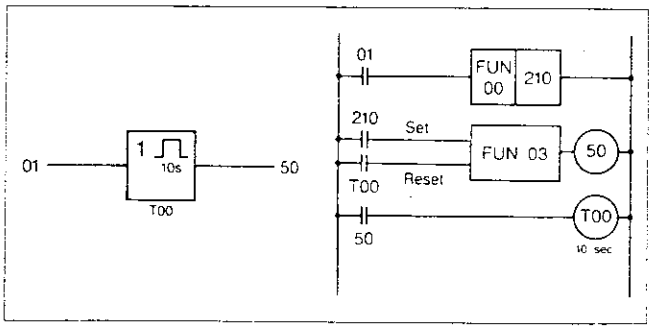


Figure 5.9

Program:

```
!          01
FUN    00 210
!          210
STR        T00
FUN        03
=          50
!          50
=      T00 010
```

If input 01 is "1", output 50 (or a flag) is set with a pulse. Output 50 also starts the timer T00. When this time has elapsed, output 50 is reset. If the input signal is short, the blocker holds itself via "FUN 03" until the set time has elapsed.

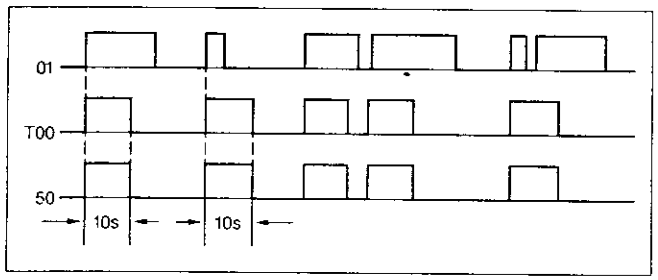


Figure 5.10

5.6 Blocker with premature termination

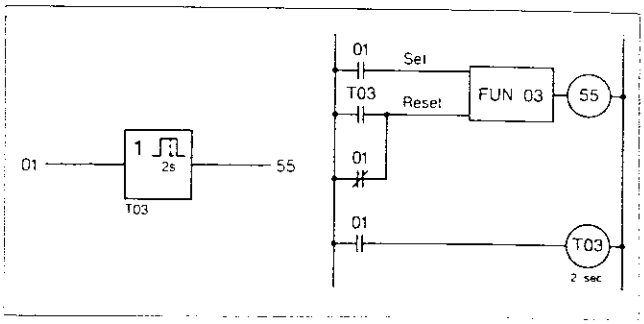


Figure 5.11

Program:

```
!          01
STR        T03
/N         01
FUN        03
=          55
!          01
=      T03 002
```

If input 01 is "1", a "1" is assigned to output 55, and timer "T03" is started. After the time has elapsed, the timer sets the output to "0". If the input signal had returned to "0" during the delay time, a "0" signal is immediately assigned to the output.

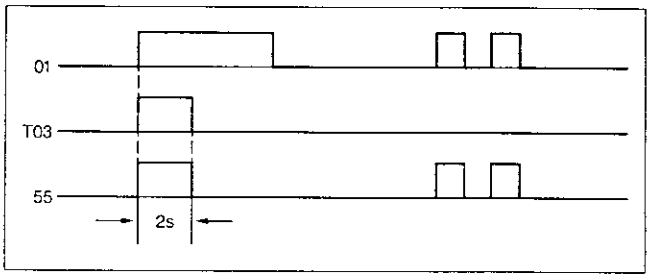


Figure 5.12

5.7 Oscillator with 1 timer

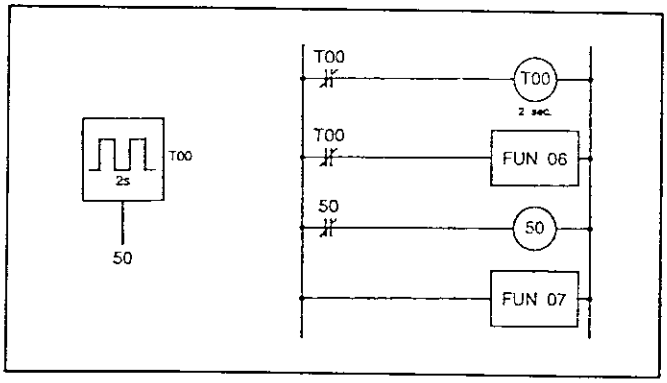


Figure 5.13

Program:

```
!N          T00
=       T00 002
!N          T00
FUN          06
!N           50
=            50
FUN          07
```

After the start of the program the time T00 will begin to run. By the instruction !NT00 with FUN 06 the program section up to FUN 07 is skipped until the time T00 has elapsed. After the expiration of the time the program section between FUN 06 and FUN 07 (!N50 = 50) is processed for the duration of one cycle and output 50 receives "1" signal. Simultaneously the time T00 is started anew, so that output 50 carries "0" signal after the expiration of the time, etc.

5.8 Oscillator with 2 timers

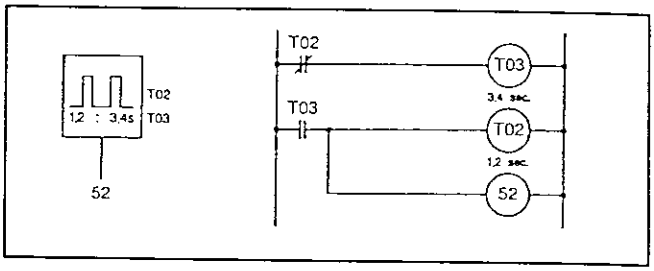


Figure 5.14

Program:

```
!N          T02
=       T03 03.4
!           T03
=       T02 01.2
=            52
```

The first time is started if output 52 is not set. When the first time (T02) has elapsed, output 52 is set. If output 52 is set, the second time (T03) is started. When this time has elapsed, output 52 is reset, and the cycle is repeated.

T02 – pulse duration
T03 – pause duration

5.9 Note for using timers

If the program cycle time exceeds 16 ms, the time base 10 ms cannot be used.

Using a basic configuration without memory expansion (950 words) the program cycle time is not longer than 16 ms, i. e. all time bases can be used without restrictions.

In case of longer programs (possible when using basic configurations with memory expansion) the program cycle time can exceed 16 ms. In this case the time base 10 ms cannot be used. The time bases 100 ms and 1 s always can be used without restrictions.

6.1 Counters general

Software counters in cycling controls have a relatively low counting frequency. This depends on the cycle time and the input delays of the signals.

With a program length of 1 K words, the cycle time in the PROCONTIC K200 is typically 5 ms, and may be even longer if FUN commands are used.

The maximum counting frequency is, taking into account the input delay (typically 4 ms),

max. 50 Hz for 1 K word program
max. 40 Hz for 2 K word program.

For rapid counting operations, it is therefore advisable to use the high speed counter (4 decades BCD, 10 kHz).

Use of counters:

Decimal counters:
For internal counting operations with a large number of setvalues or interrogations

BCD counter:
For external setvalue specification or external display of actual value.

All outputs of the counters can also be buffered in the form of flags. If the flags 400 to 767 are used, they are also buffered.

6.2 Decimal counter, up

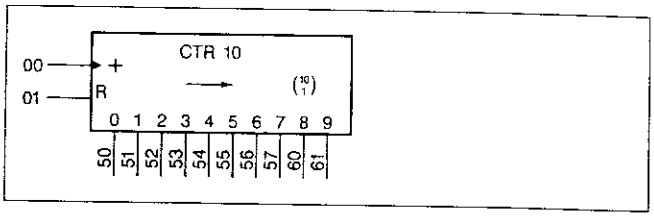


Figure 6.1

Program with FUN 47:

```
!N      51
&N      52
&N      53
&N      54
&N      55
&N      56      zero
&N      57
&N      60
&N      61
=       50
!       00
FUN  00 205
!       205
&N      61      count
=       210
!       205
&       61
/       01      reset
=       201
!       50
STR     210
STR     201
FUN  47  51
```

Each "0–1" edge at the count input 00 increments the counter by 1.

A "1" signal at input 01 resets the counter to zero.

6.3 Decimal counter, down

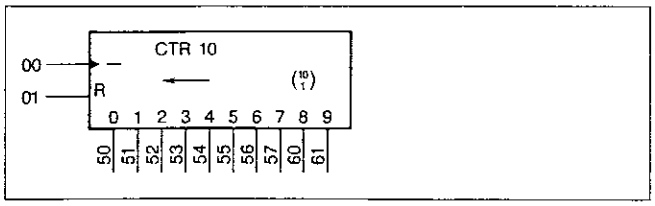


Figure 6.2

Program with FUN 47:

```
!N        51   ┐
&N        52   │
&N        53   │
&N        54   │
&N        55   │
&N        56   │  zero
&N        57   │
&N        60   │
&N        61   │
=         50   ┘
!         00
FUN   00  205
!         205  ┐
&N        51   │  count
=         210  ┘
!         205  ┐
&         51   │
/         01   │  reset
=         201  ┘
!         50
STR       210
STR       201
FUN   47  301
!         301
=         61
!         302
=         60
!         303
=         57
!         304
=         56
!         305
=         55
!         306
=         54
!         307
=         53
!         310
=         52
!         311
=         51
```

Each "0–1" edge at the count input decrements the counter by 1.

A "1" signal at input 01 resets the counter to zero.

6.4 Decimal counter, up/down, 2 decades

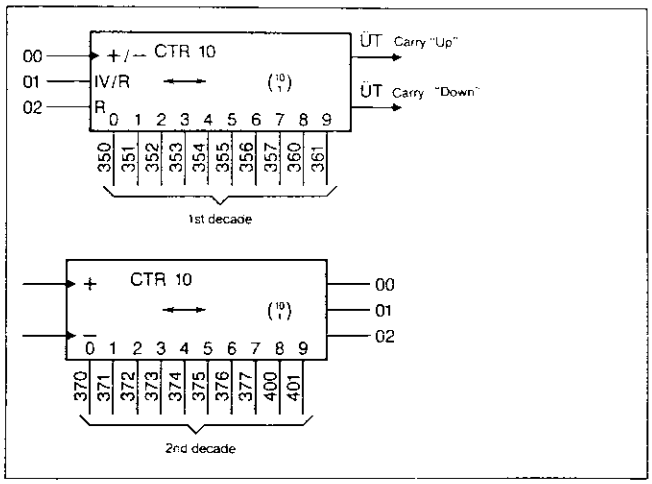


Figure 6.3

Program:

```
!          01
FUN   00  205   – Generation of pulses
!N        300  ┐
&         301  │
&         302  │
&         303  │
&         304  │
&         305  │  Carry Up
&         306  │
&         307  │
&N         00  │
=         203  ┘
!         300  ┐
&         303  │
&         304  │
&         307  │  Carry down
&          00  │
=         202  ┘
!         205
STRN      202
/N        203
&         STR
=         210
!         205
STR       202
/         203
&         STR
=         201
!          00   – U/D input
STR       210   – count
STR       201  ┐
/          02  ┘ reset
FUN   40  300
```

```
!N      300
&N      301
&N      302      Zero generation, decade 1
&N      303
=       350
!N      304
&N      305
&N      306      Zero generation, decade 2
&N      307
=       370
!       300
&N      301
&N      302
&N      303
=       351
!N      300
&       301
&N      302
&N      303
=       352
!       300
&       301
&N      302
&N      303
=       353
!N      300
&N      301
&       302      Convert decade 1 into (10/1)
&N      303
=       354
!       300
&N      301
&       302
&N      303
=       355
!N      300
&       301
&       302
&N      303
=       356
!       300
&       301
&       302
&N      303
=       357
!N      300
&N      301
&N      302
&       303
=       360
!       300
&N      301
&N      302
&       303
=       361
```

```
!       304
&N      305
&N      306
&N      307
=       371
!N      304
&       305
&N      306
&N      307
=       372
!       304
&       305
&N      306      Convert decade 2 into (10/1)
&N      307
=       373
!N      304
&N      305
&       306
&N      307
=       374
!       304
&N      305
&       306
&N      307
=       375
!N      304
&       305
&       306
&N      307
=       376
!       304
&       305
&       306
&N      307
=       377
!N      304
&N      305
&N      306
&       307
=       400
!       304
&N      305
&N      306
&       307
=       401
```

If more than 2 decades are required, the BCD counter can also be converted for 3 or 4 decades. It is only necessary to program conversion from BCD in (10/1) for each additional decade.

In the above example, input 00 is used for up/down switching.
Input 01 is the count input.
Input 02 is the reset input.

6.5 BCD counter, up

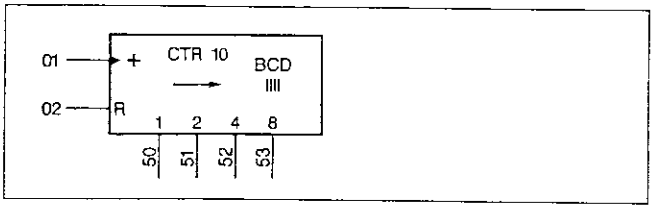


Figure 6.4

Program:

```
!          01
FUN    00 205    - Generation of count pulses
!          50
&          53      1*
=         202
!         205
&N        202
=         210
!         205
&         202
=         201
!          00    - Set U/D input to 1
STR       210    - Count input
STR       201      Reset input
/          02
FUN    40  50
```

If the record identified with 1* is replaced with the record

```
!          50
&          53
&          54
&          57
=         202
```

then the up-counter operates with 2 decades.

The outputs extend from 50 to 57, 50 being the least significant and 57 the most significant bit.

If the record identified with 1* is replaced with the record

```
!          50
&          53
&          54
&          57
&          60
&          63
=         202
```

then the counter operates with 3 decades.

The output extends from 50 to 63, 50 being the least significant and 63 the most significant bit.

The counter is incremented by 1 with each "0-1" edge at count input 01. A "1" signal at reset input 02 resets the counter to zero.

6.6 BCD counter, down

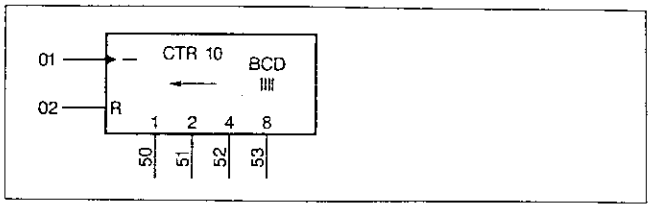


Figure 6.5

Program:

```
!          01
FUN    00 205
!N        300
&         301
&         302
&         303
=         202

①
!         205
&N        202
=         210
!         205
&         202
=         201
!          00    - U/D input with 0 signal
STR       210    - Count input
STR       201
/          02      Reset input
FUN    40 300
!         300
=          50
!         301
=          51
!         302
=          52
!         303
=          53
```

①
If & 304 & 305 & 306 & 307 is inserted before the output assignment (= 202) and the following statements are added after the last record (! 303 = 53):

```
!         304
=          54
!         305
=          55
!         306
=          56
!         307
=          57
```

then the down-counter operates with two decades. The outputs extend from 50 to 57, 50 being the least significant and 57 the most significant bit.

①

If & 304 & 305 & 306 & 307 & 310 & 311 & 312 & 313 is inserted before the following program section (before = 202), and the following statements are appended at the end of the program:

```
!   304
=    54
!   305
=    55
!   306
=    56
!   307
=    57
!   310
=    60
!   311
=    61
!   312
=    62
!   313
=    63
```

then the counter operates with three decades. The outputs extend from 50 to 63, 50 being the least significant, and 63 the most significant bit.

Each "0–1" edge at the count input 01 decrements the counter by 1. A "1" signal at the reset input 02 resets the counter to 0.

6.7 BCD counter, up/down

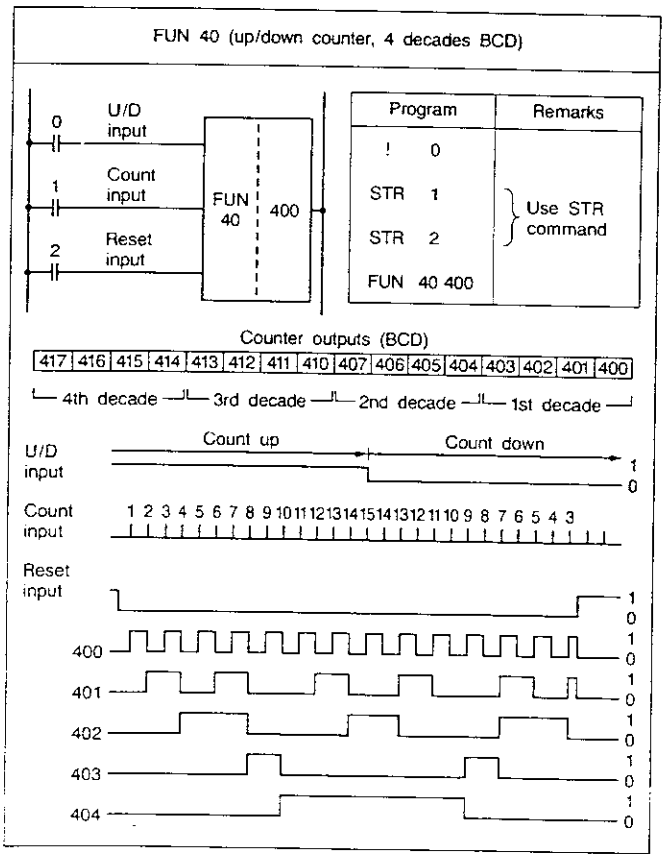


Figure 6.6

a. The up/down counters (4 decades) are programmed with "FUN 40", combined with flags, or directly with the outputs.
   If the counter value is stored in the flags 400 to 767 and then switched through to the outputs, the counter value remains stored even if voltage is interrupted. If the counter value is sent directly to the outputs, the counter is reset to 0 if the voltage fails.
b. If the counter is used as an up counter, the next pulse after the counter reaches 9999 switches the counter to 0.
c. If the counter is used as a down counter, the next pulse after 0 switches the counter to 9999.
d. The counter can be switched to 0 with "1" signal at the reset input.

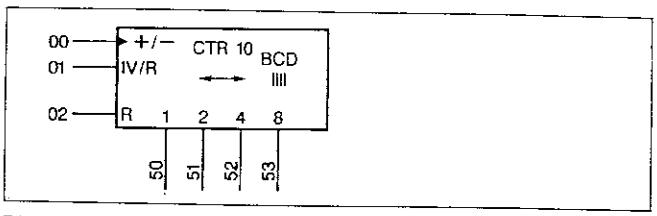


Figure 6.7

Program:

```
!          01
FUN   00  205
!N         300   ┐
&          301   │
&          302   │ up
&          303   │
&N          00   │
=          203   ┘
!          300   ┐
&          303   │ down
&           00   │
=          202   ┘
!          205
STRN       202
/N         203
&          STR
=          210
!          205
STR        202
/          203
&          STR
=          201
!           00   - U/D input
STR        210   - Count input
STR        201   ┐ Reset input
/           02   ┘
FUN   40  300
!          300
=           50
!          301
=           51
!          302
=           52
!          303
=           53
```

If the U/D counter is to operate with two decades, then the following statements must be inserted.

Up record before output assignment = 203:
& 304 & 305 & 306 & 307

Down record before output assignment = 202:
& 304 & 307

In addition, the following statements must be appended after the last program word = 53:

```
!      304
=       54
!      305
=       55
!      306
=       56
!      307
=       57
```

The outputs extend from 50 to 57, 50 being the least significant and 57 the most significant bit.

If the U/D counter is to operate with three decades, the following program sections must be inserted:

Up record before = 203:
& 304 & 305 & 306 & 307 & 310 & 311
& 312 & 313

Down record before = 202:
& 304 & 307 & 310 & 313

In addition, the following statement must be appended after the last program word = 53:

```
!      304
=       54
!      305
=       55
!      306
=       56
!      307
=       57
!      310
=       60
!      311
=       61
!      312
=       62
!      313
=       63
```

The outputs now extend from 50 to 63, 50 being the least significant and 63 the most significant bit.

6.8 BCD counters – setting

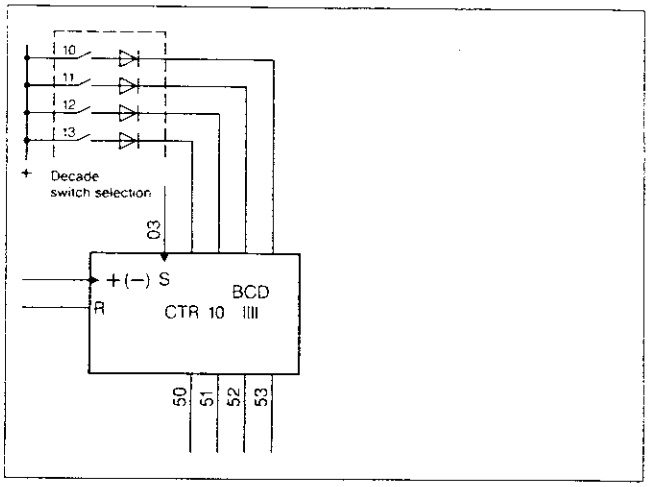


Figure 6.8

Program for the counters
– see BCD counter, up or down

Program:

```
!          03
FUN    00 230
!N         230
FUN        06
!          10
FUN        02
=          50
!          11
FUN        02
=          51
!          12
FUN        02
=          52
!          13
FUN        02
=          53
FUN        07
```

It may be necessary to insert an & function before the set input of the counter in order to block processing of count pulses during setting.

7.1 Shift register 16 bits (FUN 47)

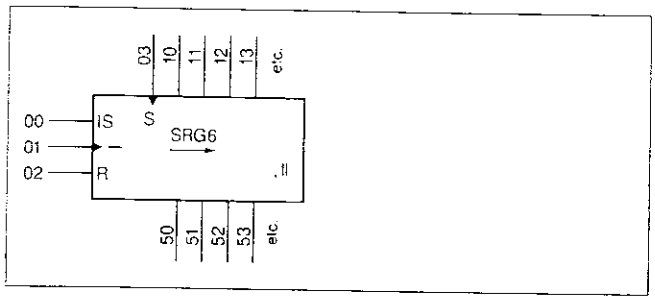


Figure 7.1

General information:

In a shift register, each bit is shifted through the shift positions (16) by one position with each clock signal. Information is loaded into the register in the first position (00) and the information from the last position is lost. For loading or correction, it is also possible to load the shift register in parallel through the inputs (10 to 27) if a pulse is present at input 03.

		50	51	52	53	etc.
		0	0	0	0	Starting condition
Inputs 03		10	11	12	13	
⎍		1 ↓	0 ↓	1 ↓	0 ↓	Parallel loading
01	00	1	0	1	0	
⎍	1	1	1	0	1	
⎍	1	1	1	1	0	
⎍	1	1	1	1	1	Shifting
⎍	0	0	1	1	1	
⎍	0	0	0	1	1	
⎍	0	0	0	0	1	
⎍	0	0	0	0	0	

The following program acts as a shift register with 16 bit. In order to simplify the program, parallel loading is provided only for the first decade. All other decades can be loaded in the same manner by means of the appropriate inputs.

Program:

```
!          01
FUN    00 200
!          00   - Serial input
STR        200  - Clock input
STR        02   - Reset input
FUN    47  50
!          03
FUN    00 201
!N         201
FUN        06
!          10
FUN        02
=          50
!          11
FUN        02
=          51
!          12
FUN        02
=          52
!          13
FUN        02
=          53
FUN        07
```

The shift register can be extended to any required number of decades.

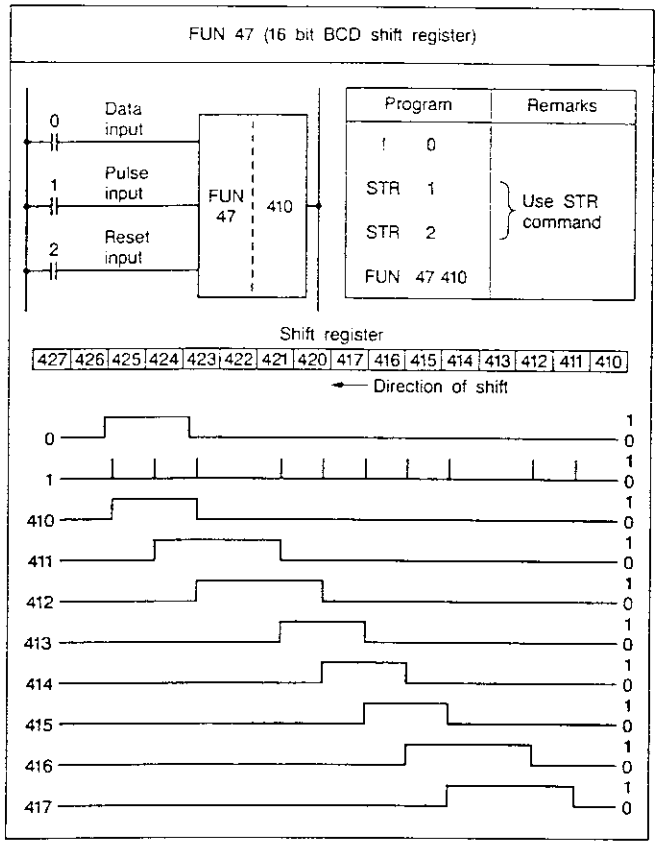


Figure 7.2

a. FUN 47 is a function for generating a serial 16 bit shift register, and is used together with the flags. If flags 400 to 767 are used, it has a remanent behavior.

b. The data at the data input are shifted from the least significant bit (in the above example: flag 410) to the most significant bit (flag 427). Instead of flags, outputs can also be programmed directly.

c. It is possible to refer to various shift registers with different flags and/or outputs (see diagram below).

7.2 FIFO register

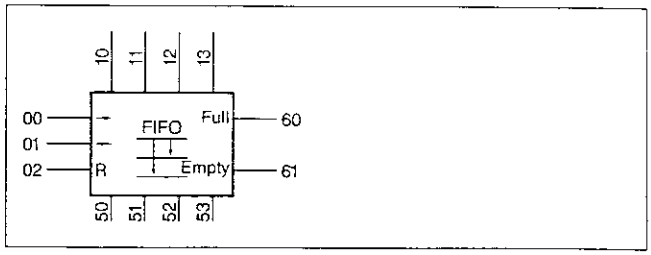


Figure 7.3

General information:

In the case of a FIFO (= first in - first out) register, information is stored in the register in parallel (10 - 13) with a signal at input 00. This information passes through the register to the first empty position or to the last position of the register if all positions are empty.

The last line of the register is erased, and all other lines are moved by one position, with the read signal at input 01. The clear signal at input 03 clears the complete register.

If the register is full, i. e. all lines occupied, no further information is accepted.

The conditions "full" and "empty" of the FIFO register are indicated.

Program:

```
!        00
&N      270      Write clock
FUN  00 200
!N      200
FUN      06
!        10
=       210
!        11
=       211
!        12
=       212
!        13
=       213
!N      270
=       270
FUN      07
!       271
FUN      06
!       210
=       220
!       211
=       221
!       212
=       222
!       213
=       223
!       270
=       271
FUN      02
=N      270
=N      210
=N      211
=N      212
=N      213
FUN      07
!       272
FUN      06
!       220
=       230
!       221
=       231
!       222
=       232
!       223
=       233
!       271
=       272
FUN      02
=N      271
FUN      07
!       273
FUN      06
!       230
=        50
!       231
=        51
!       232
=        52
!       233
=        53
```

```
!       272
=       273
FUN      02
=N      272
FUN      07
!        01      Read clock
FUN  00 274
!       274
FUN      02
=N      273
!        02
FUN      02      Clear
=N      271
=N      272
=N      273
=N      274
!N      273      Empty signal
&N      272
&N      274
=        61
!       270
=        60      - Full signal
```

Notes

8.1 Code converter, BCD into 1-out-of-10

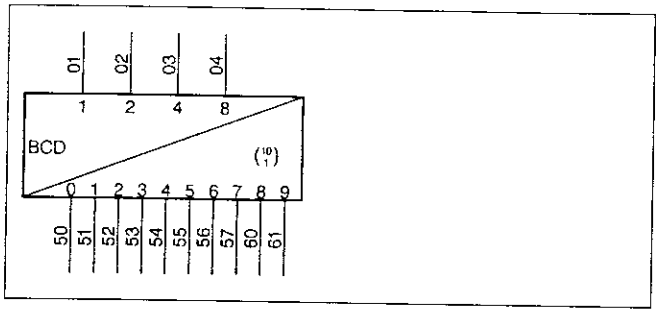


Figure 8.1

Program:

```
!N   01
&N   02
&N   03   0
&N   04
=    50
!    01
&N   02
&N   03   1
&N   04
=    51
!N   01
&    02
&N   03   2
&N   04
=    52
!    01
&    02
&N   03   3
&N   04
=    53
!N   01
&N   02
&    03   4
&N   04
=    54
!    01
&N   02
&    03   5
&N   04
=    55
!N   01
&    02
&    03   6
&N   04
=    56
```

```
!    01
&    02
&    03   7
&N   04
=    57
!N   01
&N   02
&N   03   8
&    04
=    60
!    01
&N   02
&N   03   9
&    04
=    61
```

8.2 Code converter, 1-out-of-10 into BCD

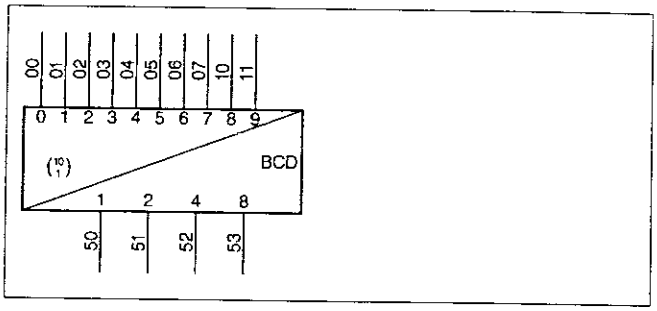


Figure 8.2

Program:

```
!    01
/    03
/    05   BCD "1"
/    07
/    11
=    50
!    02
/    03
/    06   BCD "2"
/    07
=    51
!    04
/    05
/    06   BCD "4"
/    07
=    52
!    10
/    11   BCD "8"
=    53
```

8.3 Code converter, BCD into 1-out-of-8

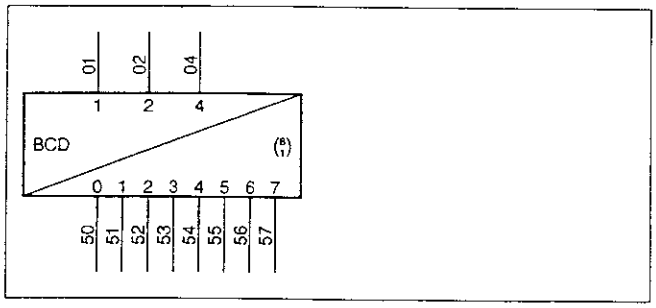


Figure 8.3

Program

```
!N    01
&N    02
&N    04
=     50
!     01
&N    02
&N    04
=     51
!N    01
&     02
&N    04
=     52
!     01
&     02
&N    04
=     53
!N    01
&N    02
&     04
=     54
!     01
&N    02
&     04
=     55
!N    01
&     02
&     04
=     56
!     01
&     02
&     04
=     57
```

0
1
2
3
4
5
6
7

8.4 Code converter, 1-out-of-8 into BCD

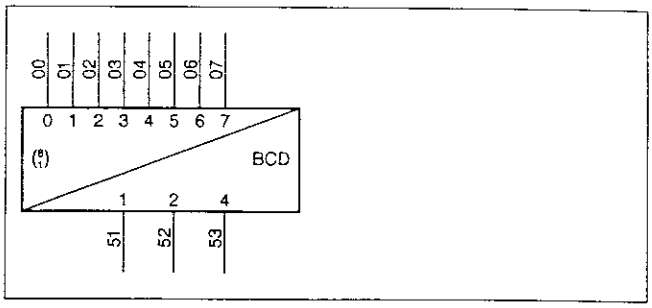


Figure 8.4

Program:

```
!     01
/     03
/     05
/     07
=     51
!     02
/     03
/     06
/     07
=     52
!     04
/     05
/     06
/     07
=     53
```

BCD "1"

BCD "2"

BCD "4"

8.5 Code converter, binary into 1-out-of-16

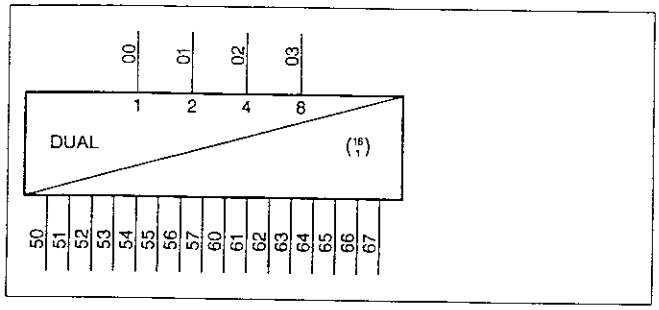


Figure 8.5

Program:

```
!N    00
&N    01
&N    02   0
&N    03
=     50

!     00
&N    01
&N    02   1
&N    03
=     51

!N    00
&     01
&N    02   2
&N    03
=     52

!     00
&     01
&N    02   3
&N    03
=     53

!N    00
&N    01
&     02   4
&N    03
=     54

!     00
&N    01
&     02   5
&N    03
=     55

!N    00
&     01
&     02   6
&N    03
=     56

!     00
&     01
&     02   7
&N    03
=     57

!N    00
&N    01
&N    02   8
&     03
=     60

!     00
&N    01
&N    02   9
&     03
=     61

!N    00
&     01
&N    02   10
&     03
=     62

!     00
&     01
&N    02   11
&     03
=     63

!N    00
&N    01
&     02   12
&     03
=     64

!     00
&N    01
&     02   13
&     03
=     65

!N    00
&     01
&     02   14
&     03
=     66

!     00
&     01
&     02   15
&     03
=     67
```

8.6 FUN 24–FUN 25, Conversion Binary (HEX) into BCD and vice versa

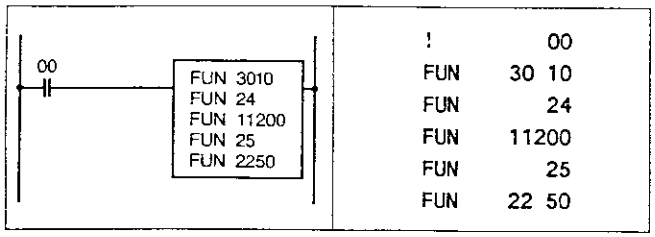


Figure 8.6

As soon as input 00 carries "1" signal (starting condition fulfilled) the status of the inputs 10 to 27 is transferred to the arithmetic register by FUN 30.

FUN 24 converts the binary value (Hex) into BCD value.

To this BCD value the value of the markers 200 to 217 is added by FUN 11, and the result is filed in the arithmetic register.

Now the value of the arithmetic register is converted to a binary value (Hex) by FUN 25, and passed on to the outputs 50 to 67 by FUN 22 (s. section 10.5).

9.1 Comparator for equivalence

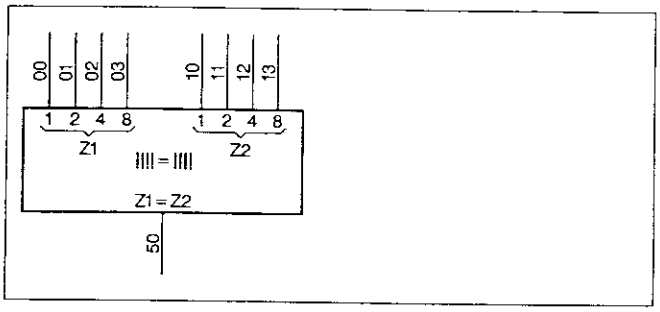


Figure 9.1

Implementation in program

Each individual bit is checked for greater or smaller. If this is not the case, the two values are equal.

Program: with STR command:

```
!N      00
&       10
STR     00
&N      10
STRN    01
&       11
STR     01
&N      11
STRN    02
&       12
STR     02
&N      12
STRN    03
&       13
STR     03
&N      13
/       STR
/       STR
/       STR
/       STR
/       STR
/       STR
/       STR
=N      50
```

Program: with flags

```
!N      00
&       10
=       200
!       00
&N      10
=       201
!N      01
&       11
=       202
!       01
&N      11
=       203
!N      02
&       12
=       204
!       02
&N      12
=       205
!N      03
&       13
=       206
!       03
&N      13
=       207
!       200
/       201
/       202
/       203
/       204
/       205
/       206
/       207
=N      50
```

If both values are equal, output 50 is activated (1 signal).

9.2 Greater-less-equal comparator (4 bit)

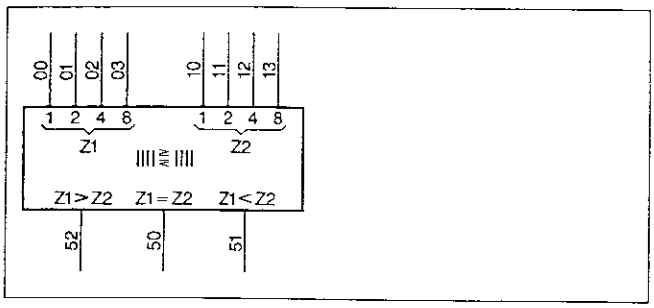


Figure 9.2

Implementation in program

Starting at the most significant bit, each original bit is checked for greater or smaller and the result connected to the appropriate output.

Program:

```
!        03
&N       13
=       200
!N       03
&        13
=       201
!N       03
&N       13
STR       3
&        13
/       STR
=       202
!        02
&N       12
&       202
=       203
!N       02
&        12
&       202
=       204
!N       02
&N       12
STR       2
&        12
/       STR
=       205
!        01
&N       11
&       202
&       205
=       206
!N       01
&        11
&       202
&       205
=       207
!N       01
&N       11
STR      01
&        11
/       STR
=       210
!        00
&N       10
&       202
&       205
&       210
=       211
!N       00
&        10
&       202
&       205
&       210
=       212
!       200
/       203
/       206
/       211
=        52
!       201
/       204
/       207
/       212
=        51
!N       52
&N       51
=        50
```

9.3 Greater-less-equal comparator (2 bit)

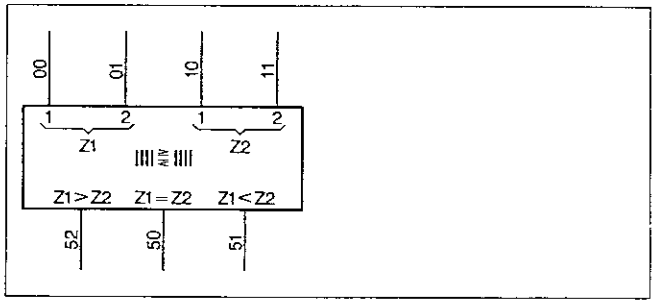


Figure 9.3

```
!        01
&N       11
=       200
!N       01
&        11
=       201
!N       01
&N       11
STR      01
&        11
/       STR
=       202
```

!	00
&N	10
&	202
=	203
!N	00
&	10
&	202
=	204
!	200
/	203
=	52
!	201
/	204
=	51
!N	52
&N	51
=	50

9.4 General information about arithmetic commands

Arithmetic commands can be processed as word (1 word = 16 bits) with PROCONTIC K200.
Arithmetic commands will be represented as shown in figure 9.4. They must be programmed always with a start condition.

The arithmetic operation (function) will be executed when the start condition has been fulfilled. By not longer fulfilled condition, the outputs or flags remain in the status they had before.

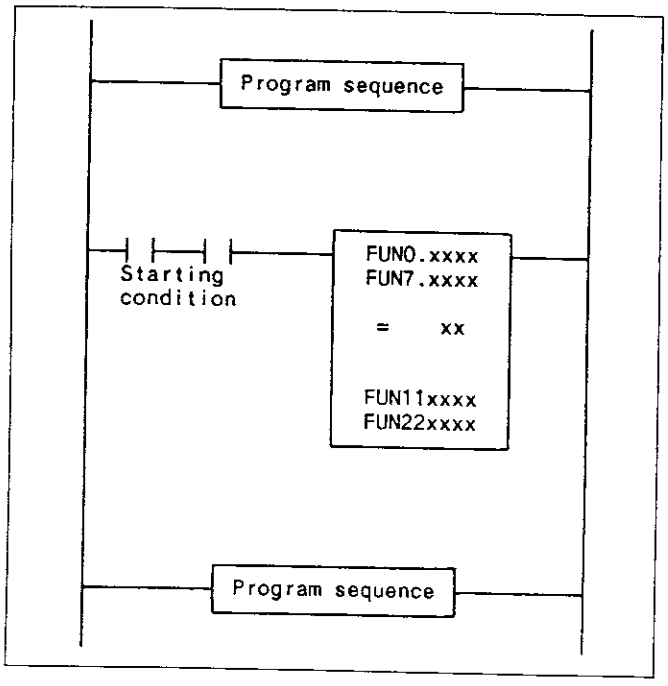


Figure 9.4

The results of arithmetic operations like addition, subtraction, multiplication, division and BNR data, will be given out only in BCD code. Arithmetic operands must be therefore converted and entered in BCD code (see section 10.5 in this description).

9.5 FUN 0. – Load constant

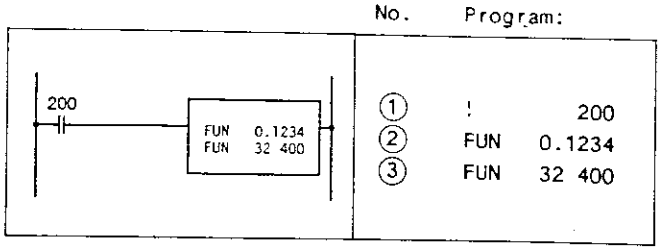


Figure 9.5

Notice to no. ① - ③:

①: ! 200 —> Start condition
②: "1234"—> AR
③: AR —> 400 - 417

The FUN 0. command will used to load a constant in BCD format in the arithmetic register.

A decimal point must be programmed before the value.

Example:

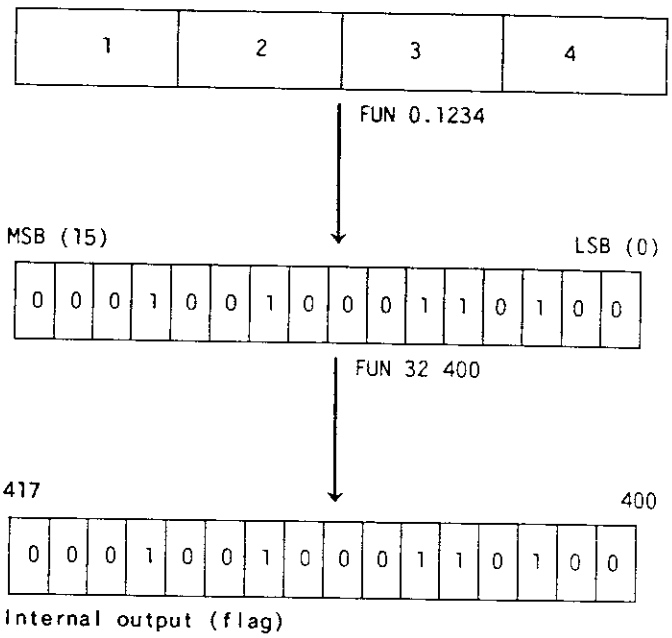


Figure 9.6

The four BCD positions data "1234" will be loaded into the arithmetic register (AR) and the AR content will be given to the buffered flags 400–417, as soon as the starting condition is fulfilled (flag 200 has "1" signal).

When the start condition (flag 200 has signal "1") is no more fulfilled (flag 200 has signal "0"), the flags remain set to the status they already had.

9.6 FUN 7. – Comparison of the constants for ≥ with the contents of the arithmetic register

Program:

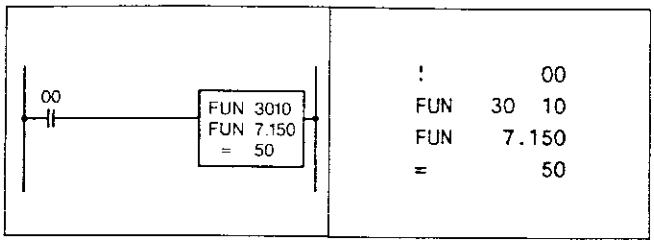


Figure 9.7

The status of the inputs 10 to 27 will be taken over in the arithmetic register, by means of FUN 30, as soon as the input 00 becomes "1" signal (start condition fulfilled). The arithmetic-register content will be compared with the constant of FUN 7. (150 in the above stated example).

The output 50 becomes signal "1" when the arithmetic-register content is larger or equal to (≥) the constant of function FUN 7.

The output 50 remains in idle condition ("0" signal) when the arithmetic-register content is smaller than (<) the constant of function FUN 7.

Note: By time comparisons with FUN 7., the constant will be entered in 0.1 s steps (for instance, insertion 600 means 60 s).

9.7 FUN 8. – Comparison of the constants for = with the contents of the arithmetic register

Program:

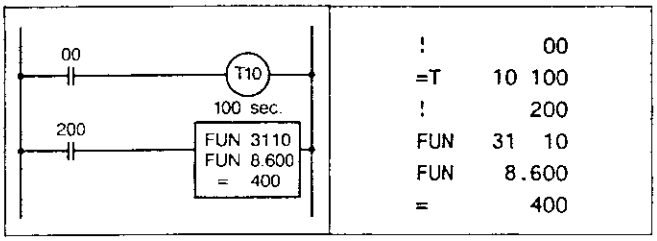


Figure 9.8

The instantaneous value of the T10 timer will be taken over in the arithmetic register with FUN 31, as soon as the start condition (flag 200 has "1" signal) has been fulfilled. The arithmetic register content will be compared with the constant of FUN 8 (600 in this example).

Flag 400 is set to "1" signal when the arithmetic-register content is equal (=) to the constant of FUN 8.

The flag 400 remains in idle condition ("0" signal) when the arithmetic-register content is not equal to the constant of FUN 8.

Note: By time comparisons with FUN 8. (600 in the above shown example) the constant will be entered in 0.1 s steps, (600 x 0.1 s = 60 s).

9.8 FUN 9. – Comparison of the constants for < with the contents of the arithmetic register

Program:

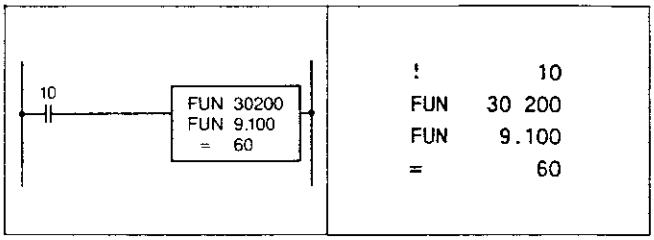


Figure 9.9

The status of the flags 200 to 217 will be taken over in the arithmeticregister with FUN 30, as soon as input 10 becomes "1" signal (start condition fulfilled).

The arithmetic-register content will be compared with the constants of FUN 9. (100 in this example).

The output 60 becomes "1" signal when the arithmetic-register content is smaller (<) than the constant of FUN 9.

The output 60 remains in idle condition ("0" signal), when the arithmetic register content is larger or equal (≥) to the constant.

Note: By time comparisons with FUN 9., the constant will be entered (100 in the above shown example) in 0.1 s steps:
(100 x 0.1 s = 10 s).

9.9 Summary example for the functions FUN 7., FUN 8., FUN 9.

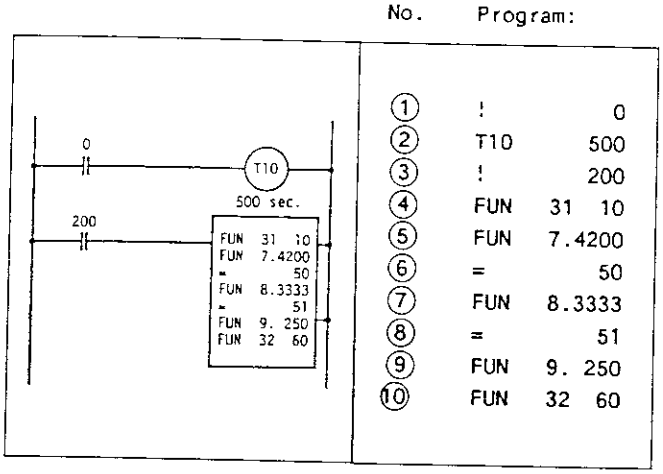


Figure 9.10

Notice to no. ④ - ⑩ :

④ : Real value of T10 --> AR
⑤ : AR ≥ "4200" ... 1 --> C
⑥ : If C = 1, then 50 on
⑦ : AR = "3333" ... 1 --> C
⑧ : If C = 1, then 51 on
⑨ : AR < "250" ... 1 --> C
⑩ : If C = 1, AR --> (60-77)
If C = 0, 0 --> (60-77)

If the start condition is fulfilled (flag 200 has "1" signal), then the actual value of the T10 timer will be transferred to the (AR) arithmetic register with FUN 31 and compared with the constants of FUN 7., FUN 8. and FUN 9.

The actual value of T10 will be 4035 (403.5 s) in the next explanations.

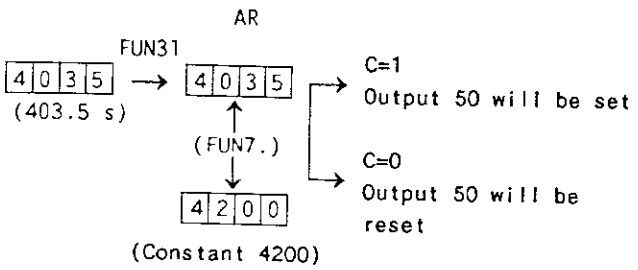


Figure 9.11

The actual T10 value (4035) will be loaded into the arithmetic register (AR) with FUN 31. From the comparison with the constant of FUN 7. (4200) results that the value in AR is smaller than the constant of FUN 7. (4035 < 4200). Thereby the carry flag C will be reset to 0, which in turn, resets the output 50.

The comparison result can be given out in bit format. The on/off status of the carry flag C will be given out together with the result.

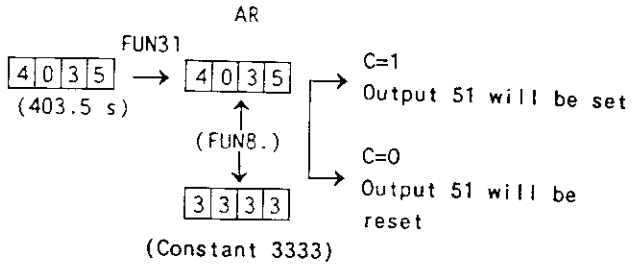


Figure 9.12

The actual T10 value (4035) will be loaded into the arithmetic register (AR) with FUN 31. From the comparison with the constant of FUN 8. (3333) results that the value in AR is not equal to the constant of FUN 8. Thereby the carry flag C will be reset to 0, which in turn, resets the output 51.

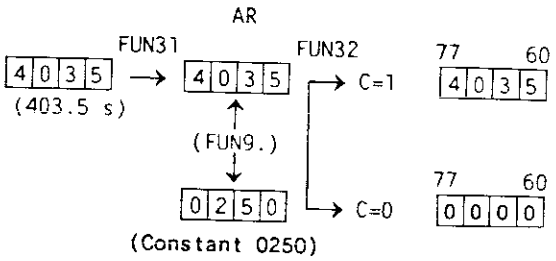


Figure 9.13

The actual T10 value (4035) will be loaded into the arithmetic register (AR) with FUN 31. From the comparison with the constants of FUN 9. (0250) results that the value in AR is larger than the constant of FUN 9. Thereby the carry flag will be reset to 0, which in turn, brings the word 0000 to the outputs 60 to 77 with FUN 32.

9.10 Carry flag status by the comparison commands

The following table shows the value of the carry flag C, depending if the corresponding constant of the comparison values FUN 7., FUN 8. and FUN 9. is larger, equal or smaller than the arithmetic register actual value.

Command	AR>Const.	AR=Const.	AR<Const.
FUN 7. $\geq$ AR	C = 1	C = 1	C = 0
FUN 8. = AR	C = 0	C = 1	C = 0
FUN 9. < AR	C = 0	C = 0	C = 1

For the average execution time of the calculation commands please refer to the table on page 1–5 of this ABB Procontic K200 software description.

10.1 FUN 11 – Addition

Example 1:

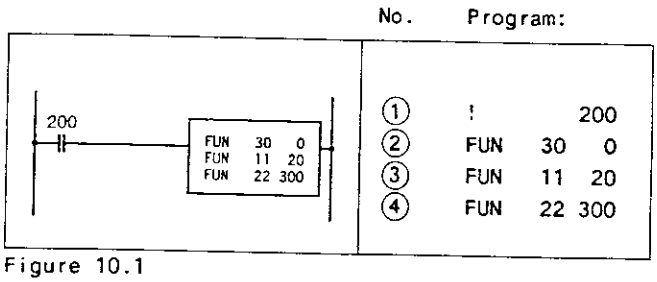


Figure 10.1

Notice to no. ① – ④:

①: ! 200 —> Start condition
②: 0 – 17 —> AR
③: AR + (20–37) —> AR
④: AR —> 300–317

The command FUN 11 will be used when something is to be added to the AR content.

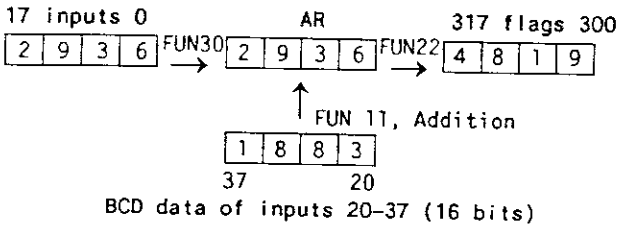


Figure 10.2

In the above example, the BCD data of inputs 0 – 17 will be taken over with FUN 30 into the arithmetic register (AR), when the start condition (flag 200 having signal "1") is fulfilled.

The BCD value, present at the inputs 20 – 37 will be added to the AR content, with FUN 11 and the new value will be stored in the AR.

The result will be given to the flags 300 – 317 with FUN 22.

Restriction: A carry flag will be set by a result larger than 9999. It will be further worked with the value to which something is to be added (the value present at inputs 10 – 17 in this example), which means that the value will be given unprocessed to the flags.

The AR content will be given to the outputs or flags (word output) with FUN 22.

Note: As soon as the AR has been read out, a new start condition is to define for a new access to the AR.

The difference between FUN 32 and FUN 22 is represented in the following table:

Command	Carry-Flag Status	Function
FUN 32	C = 0	No data out to flags/ nor outputs
	C = 1	AR content to flags/ outputs
FUN 22	C = 0 or C = 1	AR content to flags/ outputs

Example 2:

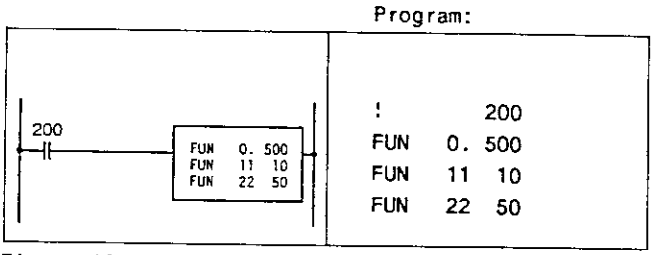


Figure 10.3

If the start condition is fulfilled, (flag 200 has signal "1") then the value of the constant (500 in this example) will be taken over into the arithmetic register, with FUN 0. With FUN 11 the value of inputs 10 to 27 will be added to this value and will be stored in the arithmetic register.

The arithmetic-register value will be given to the outputs 50 to 67, with FUN 22.

Restriction: If the result of the addition of two figures (for instance 3350 + 7000) is > 9999, then the figure to which something is to be added, will be taken into account.

10.2 FUN 12 – Subtraction

Example 1:

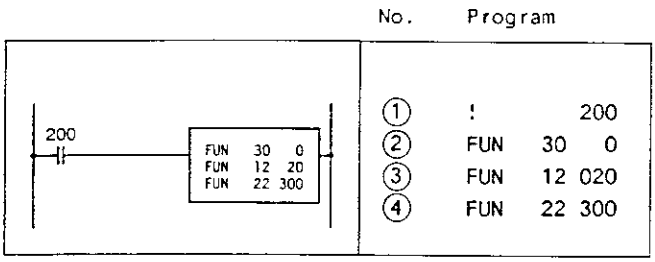


Figure 10.4

Notice to no. ① – ④:

①: ! 200 —> Start condition
②: 0 – 17 —> AR
③: AR minus (20–37) —> AR
④: AR —> 300 – 317

The command FUN 12 will be used when something is to be subtracted from the arithmetic register (AR).

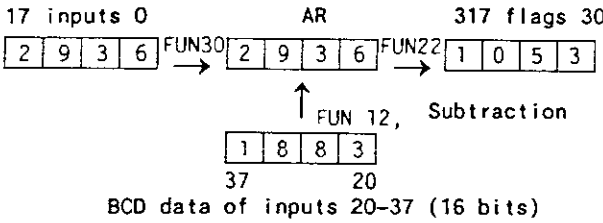


Figure 10.5

The BCD data present at inputs 0 – 17 in the above example, will be taken over into the arithmetic register with FUN 30, when the start condition (flag 200 has signal "1") is fulfilled.

The BCD value, present at the inputs 20 – 37 will be subtracted with FUN 12 from the AR content and the new value will be stored in the AR. The result will be given with FUN 22 to the flags 300 – 317.

Restriction: A carry flag C will be set in case of negative result. It will be worked with the value from which something is to be subtracted (with the value present at inputs 10 – 17 in this example), which means that the value will be given unprocessed to the flags 300 – 317, with FUN 22.

Example 2:

Program:

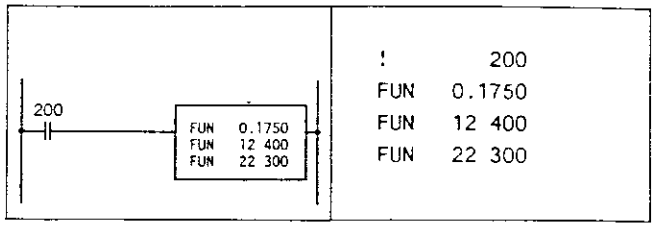


Figure 10.6

As soon as the start condition is fulfilled (flag 200 has "1" signal), the value of the constant (1750 in this example) will be taken over into the arithmetic register, with FUN 0.

The status of flags 400 to 417 will be subtracted from this value, with FUN 12, and will be stored in the arithmetic register.

Restriction: The representation of negative results is not possible (for inst. 200 – 300). In case of negative result, the figure from which something is to be subtracted, will be taken into account in the program.

10.3 FUN 13 – Multiplication

Example 1:

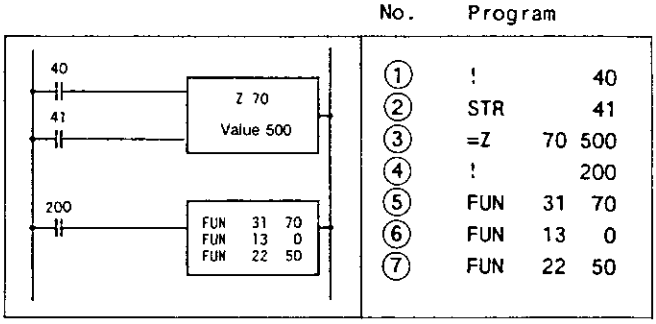


Figure 10.7

Notice to no. ④ – ⑦:

④: ! 200 —> Start condition
⑤: Z70 —> AR
⑥: AR x (0 – 17) —> AR
⑦: AR —> 50 – 67

The command FUN 13 will be used when the arithmetic-register value (AR) is to be multiplied by another BCD value.

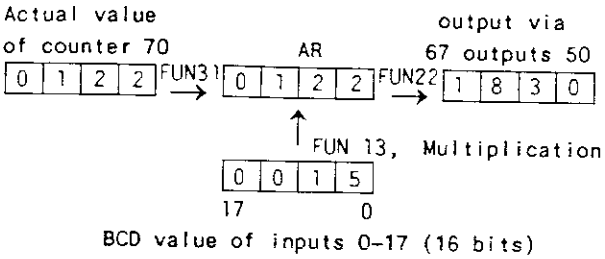


Figure 10.8

The actual counter status in the above example will be taken over with FUN 31 into the arithmetic register (AR), when the start condition (flag 200 has signal "1") is fulfilled.

With FUN 13, the actual value, present at the inputs 0 - 17 will be multiplied by the the BCD value of the arithmetic register and will be stored in the AR. With FUN 22 the AR value will be given to the outputs 50 - 67.

Restriction: A carry flag C will be set in case of a result larger than 9999. It will be further worked with the value which should have to be multiplied by a figure. It means that with FUN 22 the value will be given unprocessed to the outputs 50 - 67.

Example 2:

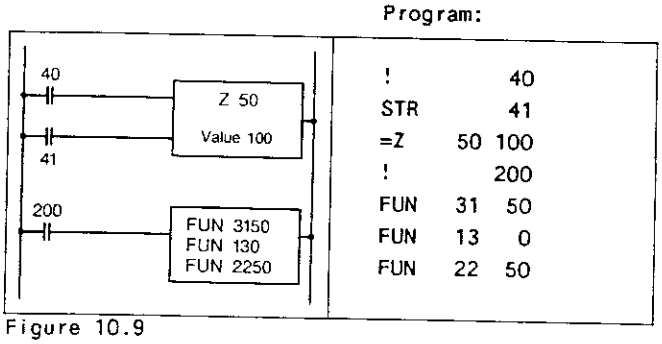


Figure 10.9

As soon as the start condition (flag 200 has signal "1") is fulfilled, the real value of counter 50 will be taken over into the arithmetic register, with FUN 31. This value will be multiplied by the status of inputs 0 to 17 (FUN 13) and will be stored in the arithmetic register.

The arithmetic-register value will be given to the outputs 50 to 67, with FUN 22.

Restriction: If the result of the multiplication of two figures is > 9999 (for example 900 x 700), then the figure will be taken into account further in the program, by which a figure should be multiplied.

10.4 FUN 14 - Division

Example 1:

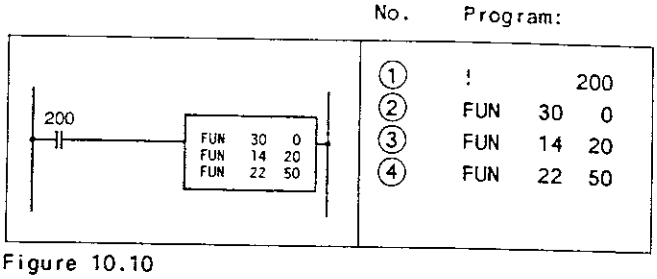


Figure 10.10

Notice to No. ① - ④:

① : ! 200 --> Start condition
② : 0 - 17 --> AR
③ : AR : (20-37) --> AR
④ : AR --> 50-67

The command FUN 14 is used to divide the arithmetic register value (AR) by another BCD value.

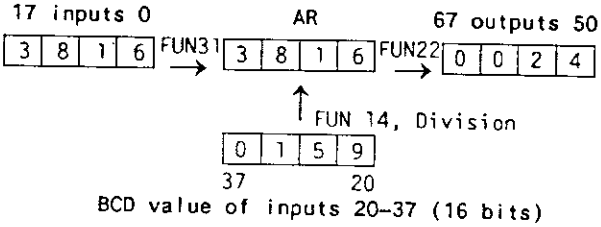


Figure 10.11

In the above example, the BCD data of inputs 0 - 17 will be taken over into the arithmetic register (AR) with FUN 30, when the start condition (flag 200 has "1" signal) is fulfilled.

The BCD value of the AR will be divided by the actual value, present at the inputs 20 - 37, with FUN 14, and will be stored in the AR. The AR value will be given to the outputs 50 - 67, with FUN 22.

Example 2:

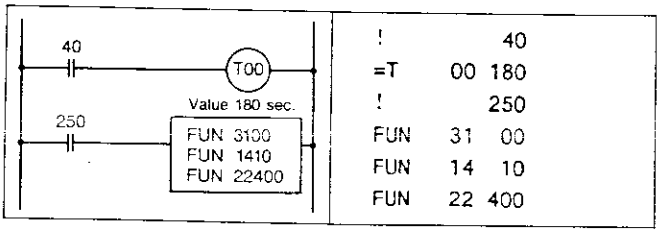


Figure 10.12

As soon as the start condition (flag 200 has "1" signal) has been fulfilled, the current value of timer T00 will be taken over into the arithmetic register with FUN 31.

This value will be divided by the status of inputs 10 to 27 with FUN 14, and will be stored in the arithmetic register.

The arithmetic register value will be given to the flags 400 to 417 with FUN 22.

Restrictions:

a) A carry flag C will be set in case of a division by 0. It will be further worked with the value which should have to be divided by a figure. It means that the value will be given unprocessed to the outputs 50–67 with FUN 22.

b) A round down operation will be carried out in case that the result of a division should contain a decimal part (for instance 19:8 = 2.275 will be rounded down to 2).

10.5 FUN 24 – FUN 25, Conversion BNR – BCD and BCD – BNR

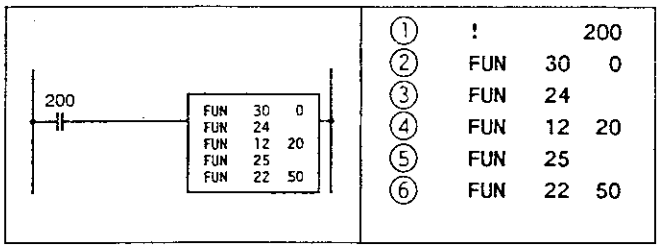


Figure 10.13

Notice to no. ① to ⑥:

①: ! 200 → Start condition
②: 0–17 → AR
③: AR (BNR) — AR (BCD)
④: AR (20 – 37) → AR
⑤: AR (BCD) — AR (BNR)
⑥: AR — (50 – 67)

The command FUN 24 will be used to convert binary data in BCD data.

The command FUN 25 will be used to convert BCD data in binary data.

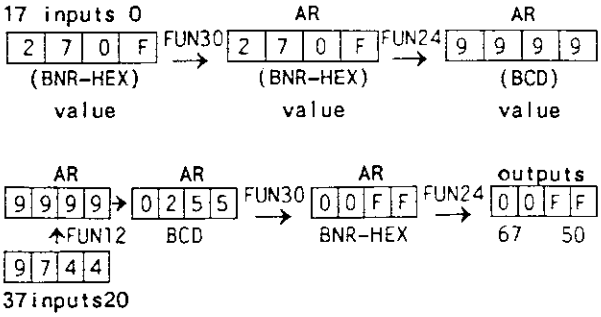


Figure 10.14

No conversion takes place when the BCD value by a conversion is larger than 9999. The carry flag C will be set.

10.6 Carry flag status by calculation commands

The following table shows the status of the carry flag C by the calculation operations.

Command	AR or insertion value		Result		
	BNR	BCD	<0	0–9999	>9999 or :0
FUN 11 ADD	C=0	C=0	–	C=0	C=1
FUN 12 SUB	C=0	C=0	C=1	C=0	–
FUN 13 MUL	C=1	C=0	–	C=0	C=1
FUN 14 DIV	C=1	C=0	–	C=0	C=1
FUN 24 BNR–BCD	C=0	–	–	C=0	C=1
FUN 25 BCD–BNR	C=1	C=0	–	–	–

Please refer to the table on page 1–5 in this ABB Procontic K200 software description for the average command execution time for calculating operations.

10.7 Programming example for an arithmetic equation

Assignation of analog input/output modules:

The following figure shows the configuration of a basic ABB Procontic K200 central unit, which has been expanded with two analog modules.

Basic central units 07 KR 220, 07 KR 228 07 KT 228, 07 KR 240 07 KR 264	07 EA 200 Analog input module	07 AA 200 Analog output module

Figure 10.15

Input Channel 0

100–107	110–117
Analog value	unused

Output Channel 1

160–167	170–177
Analog value	unused

Arithmetic equation: Y = X • A : B

X: Analog input, assigned to channel 0
Y: Analog output, assigned to channel 1
A: Constant, value: 5
B: Constant, value: 25

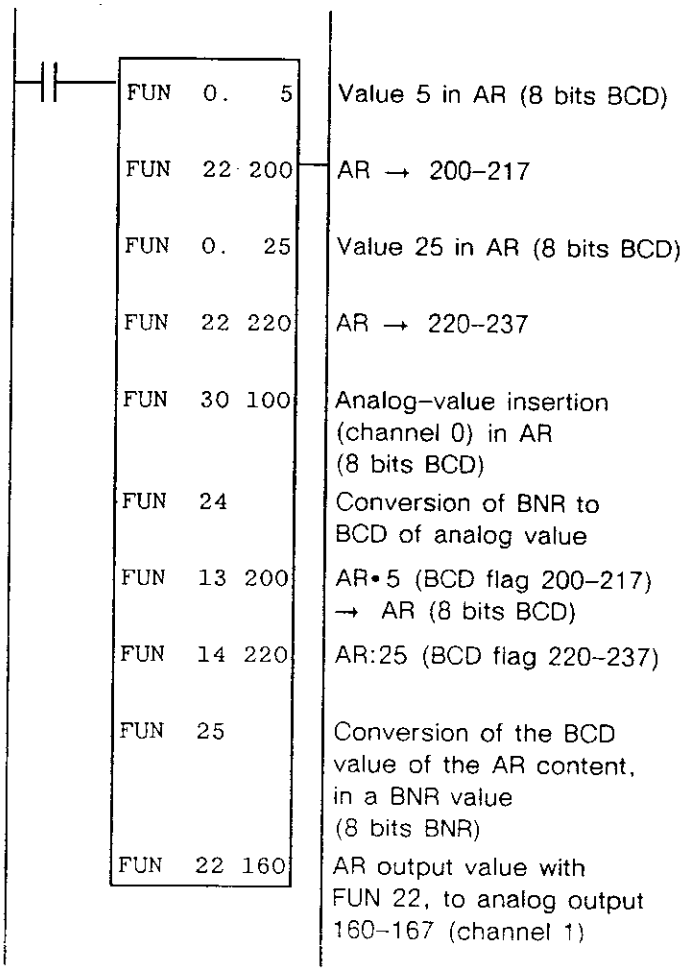


Figure 10.16

10.8 3 points regulator with hysteresis

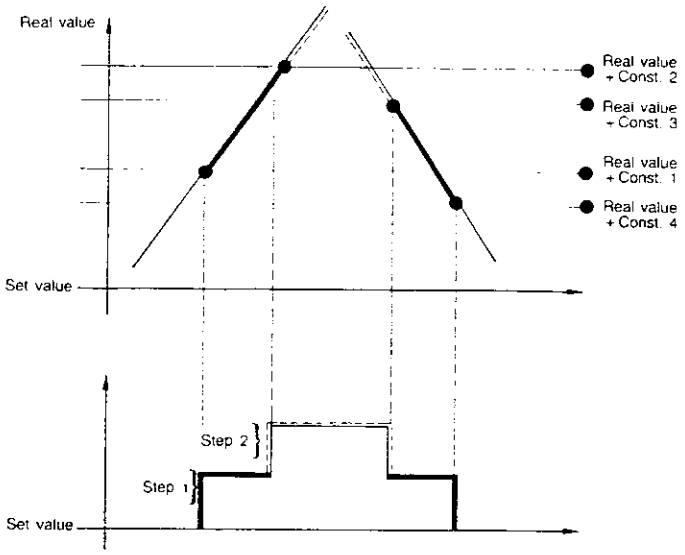


Figure 10.17

Flag zone 1	Real value	Flags 300–317
Flag zone 2	Set value	Flags 320–317
Flag zone 3	Set value + Constant 1	Flags 340–357
Flag zone 4	Set value + Constant 2	Flags 360–377
Flag zone 5	Set value + Constant 3	Flags 240–257
Flag zone 6	Set value + Constant 4	Flags 260–277

Enable = Start condition = Input 2

Program:

```
!          2      Start value
FUN  30  100      Read real analog value in AR
FUN  24           BNR–BCD conversion
FUN  22  300      Real BCD value

!          2      Start condition
FUN  30  120      Read set analog value in AR
FUN  24           BNR–BCD conversion
FUN  22  320      BCD set value

!          2      Start condition
FUN  0.   40      Constant 1
FUN  11  320      Addition set value
                  + constant 1
FUN  22  340      Set value + constant 1
```

```
!          2        Start condition
FUN  0.    50       Constant 2
FUN  11    320      Addition set value + constant 2
FUN  22    360      Set value + constant 2

!          2        Start condition
FUN  0.    30       Constant 3
FUN  11    320      Addition set value + constant 3
FUN  22    240      Set value + constant 3

!          2        Start condition
FUN  0.    45       Constant 4
FUN  11    320      Addition set value + constant 4
FUN  22    260      Set value + constant 4

!          2        Start condition
FUN  30    340      Result set value + constant 1
FUN  34    300      Comparison with real value
=          400      Flag 400 "1" signal, if real
                    value ≥ set value + constant 1
!          2        Start condition
FUN  30    360      Result set value + constant 2
FUN  34    300      Comparison with real value
=          401      Flag 401 "1" signal, if real
                    value ≥ set value + constant 2

!          2        Start condition
FUN  30    300      Real BCD value
FUN  34    240      Comparison with set value
                    + constant 3
=          402      Flag 402  "1" signal, if real
                    value ≤ set value + constant 3

!          2        Start condition
FUN  30    300      BCD real value
FUN  34    260      Comparison with set value
                    + constant 4
=          403      Flag 403 "1" signal if real
                    value ≤ set value + constant 4
```

Step 1 will be set by means of flag 400, with the RS flip-flop (FUN 03), and will be reset by means of flag 403.

```
!          400      Set (turn on)
STR        403      Reset (turn off)
FUN         03      R/S latch
=           50
```

Step 2 will be set by means of flag 401, with the RS flip-flop (FUN 03), and will be reset by means of flag 402.

```
!          401      Set (turn on)
STR        402      Reset (turn off)
FUN         03      R/S latch
=           51
```

Flag 400 will have "1" signal as soon as the real value reaches or overreaches the level (set value + constant 1).

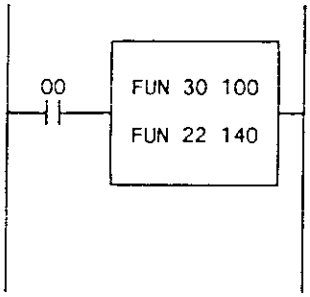
Flag 401 will have "1" signal as soon as the real value reaches or overreaches the level (set value + constant 2).

Flag 402 will have "1" signal as soon as the real value reaches or underreaches the level (set value + constant 3).

Flag 403 will have "1" signal as soon as the real value reaches or underreaches the level (set value + constant 4).

10.9 Read in/out analog values

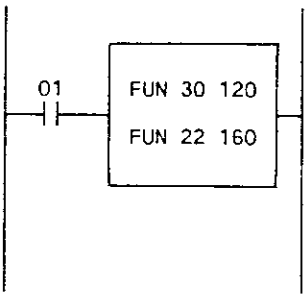
Example 1: Read in an analog value from analog input (channel 0) and read out the analog value on analog output channel 0.



Program		Remark
!	00	Starting condition
FUN 30	100	Loading of analogous value to AR
FUN 22	140	Output of analogous value

Figure 10.18

Example 2: Read in an analog value from analog input (channel 1) and output analog value on analog output channel 1.

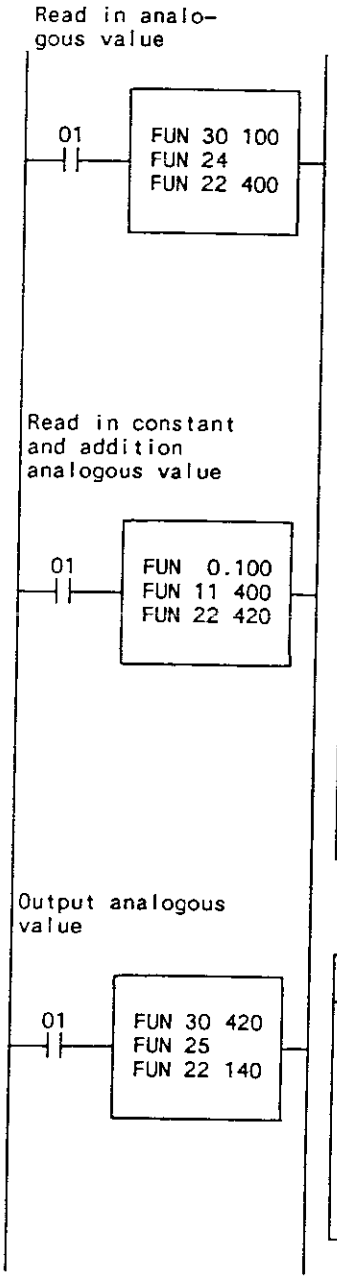


Program		Remark
!	01	Starting condition
FUN 30	120	Loading of analogous value to AR
FUN 22	160	Output of analogous value

Figure 10.19

10.10 Addition (BCD) constant to analog value – Output the new value as analog value

A constant in BCD format will be added to an analog value, in the following example. The resulting value from this operation will be given out as an analog value at an analog output.



Program	Remark
! 01	Starting condition
FUN 30 100	Loading of analogous value to AR
FUN 24	Conversion BNR–BCD
FUN 22 400	Analogous value stored in BCDformat

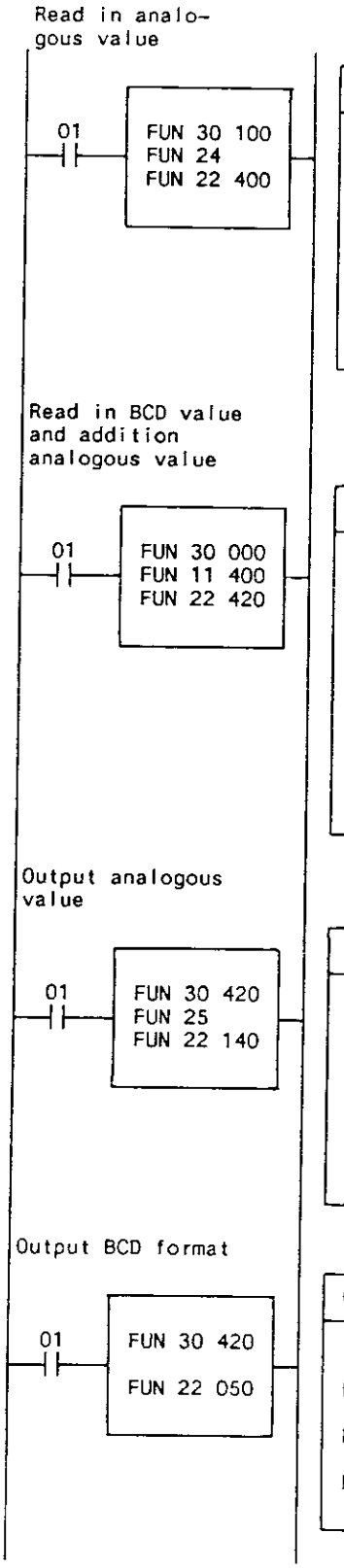
Program	Remark
! 01	Starting condition
FUN 30 000	Loading of constant into AR, BCD
FUN 11 400	Addition of analogous value in BCD format
FUN 22 420	Sum of analogous value and constant

Program	Remark
! 01	Starting condition
FUN 30 420	Loading of sum into AR
FUN 25	Conversion BCD–BNR
FUN 22 140	Output of analogous value

Figure 10.20

10.11 Addition of BCD value to analog value – Value output at analog output, or respectively, output in BCD format

A BCD value will be added to an analog value, in the following example. The result will be given at an analog output or respectively, in BCD format.



Program	Remark
! 01	Starting condition
FUN 30 100	Loading of analogous value to AR
FUN 24	Conversion BNR–BCD
FUN 22 400	Analogous value stored in BCDformat

Program	Remark
! 01	Starting condition
FUN 30 000	Loading of constant into AR, BCD
FUN 11 400	Addition of analogous value in BCD format
FUN 22 420	Sum of analogous value and constant

Command	Remark
! 01	Starting condition
FUN 30 420	Loading of sum into AR
FUN 25	Conversion BCD–BNR
FUN 22 140	Output of analogous value

Command	Remark
! 01	Starting condition
FUN 30 420	Loading of sum into AR
FUN 22 050	Output of value to
FUN 22 140	Outputs 50 to 67

Figure 10.21

10.12 Limitation of an analog value to an upper limit (for instance 8 V)

a) After successful start condition (input 01 has "1" signal), the analog value will be read in the AR, with FUN 30. The value will be converted with FUN 24 in BCD format. The converted analog value will be given out, with FUN 22, in BCD format (to the flags 300-317 in this example).

```
!        01      Start condition
FUN  30 100      Read in analog value
FUN  24          BNR to BCD conversion
FUN  22 300      Output to flags 300-317
```

b) The converted analog value will be read in again in the AR with FUN 30. It occurs then the comparison with the constant 205, which corresponds to 8 Volts (255=10 Volts). No analog output takes place when the value is ≥ 8 Volts. Output 50 will be set.

```
!        01      Start condition
FUN  30 300      Read in converted analog value
FUN  7. 205      Comparison for ≥ with constant
                 205 (corresponds to 8 Volts)
=        50      Output 50
```

c) Program module for analog value smaller than 8 Volts: As soon as the analog value is smaller than 8 Volts, the read in analog value will be given out.

```
!        50      Value is ≤ 8 Volts
FUN  06          Jump to the beginning
!        01      Start condition
FUN  30 300      Read in converted analog value
FUN  25          BCD-BNR conversion
FUN  22 140      Output analog value
FUN  07          Jump to the end
```

d) Program module for analog value larger than 8 Volts: As soon as the analog value is larger than 8 Volts, the value of the constant 205 (corresponds to 8 V) will be converted with FUN 25 (BCD-BNR) and given out.

```
!N       50      Value is ≥ 8 Volts
FUN  06          Jump to the beginning
!         1      Start condition
FUN  0. 205      Load constant 205 (8 V)
FUN  25          BCD-BNR conversion
FUN  22 140      Analog value output limited to 8 V
FUN  07          Jump to the end
```

10.13 Limitation of an analog value to upper and lower limit (upper limit 8 V, lower limit 2 V)

a) After successful start condition (input 01 has "1" signal), the analog value will be read in the AR with FUN 30. The value will be converted in BCD format, with FUN 24. The converted analog value will be given out in BCD format with FUN 22 (to the flags 300-317 in this example).

```
!        01      Start condition
FUN  30 100      Read in analog value
FUN  24          BNR to BCD conversion
FUN  22 300      Output to flags 300-317
```

b) The converted analog value will be read in the AR with FUN 30. The value will be converted to BNR format with FUN 25. The output occurs with FUN 22, on the analog output.

```
!        01      Start condition
FUN  30 300      Raed in converted analog value
FUN  25          BCD-BNR conversion
FUN  22 140      Analog output
```

c) It occurs a comparison with the constant 205 (upper limit 8 V). The output 52 will be set, if the value is ≥ 8 Volts.

```
!        01      Start condition
FUN  30 300      Read in converted analog value
FUN  7. 205      Compare constant 205 (8 V)
=        52      Output 52 will be set if the
                 analog value is ≥ 8 V
```

d) A comparison is made only with the constant 052 (lower limit 2 Volt). The output 50 will be set, if the value is ≤ 2 V.

```
!        01      Start condition
FUN  30 300      Read in converted analog value
FUN  9. 052      Compare constant 052 (2 V)
=        50      Output 50 will be set, if the
                 analog value is ≤ 2 V
```

e) Interrogation of the analog value within the 2-8 Volts zone.

```
!N       50      Analog value is smaller than 2 Volts
&N       52      Analog value is larger than 8 Volts
=        51      Analog value is within the 2-8 Volts
                 zone.
```

11.1 Signalling with continous light

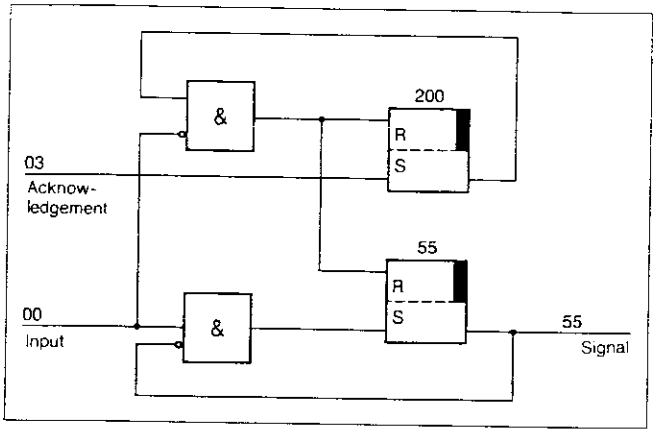


Figure 11.1

The signal at input 00 which is to be signalled is indicated by continous light at the output (55). If the condition still exists when acknowledged at input 03, the continous light remains on until the condition disappears. Otherwise, the lamp is switched off immediately after acknowledgement.

Function diagram

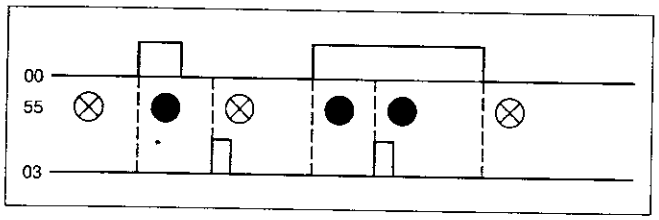


Figure 11.2

Program:

```
!       00
&N      55
FUN     02
=       55
!       03
FUN     02
=       200
!       200
&N      00
FUN     02
=N      55
=N      200
```

11.2 New value signalling with single flashing light

Each new signal at input 00 is indicated by flashing with 0.5 Hz at output 54 (FUN 774). At the same time, an audible signal is generated at output 50. If the acknowledgement key 01 is pressed, the flashing light switches to a continous light. If the condition no longer exists, the indicator at output 54 is switched off. The audible signal can be switched off only with its own acknowledgement key at input 02.

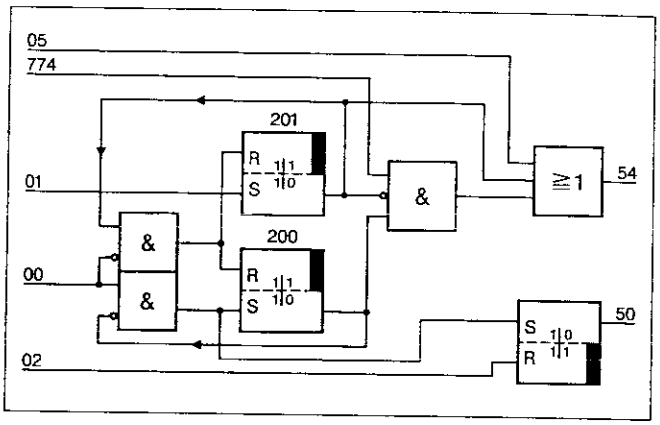


Figure 11.3

Function diagram

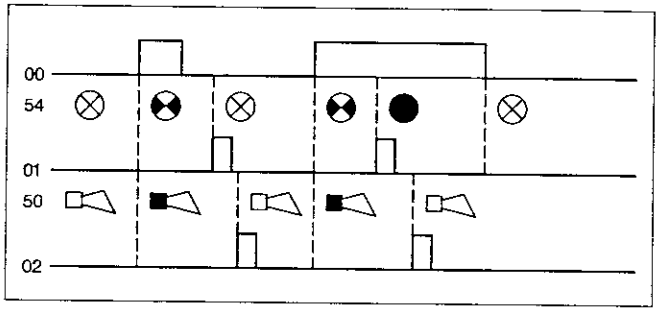


Figure 11.4

Program:

```
!        00
&N      200
FUN      02
=       200
FUN      02
=        50
!        01
FUN      02
=       201
!N       00
&       201
FUN      02
=N      200
FUN      02
=N      201
!       200
&N      201
&       774
/       201
/        05
=        54
!        02
FUN      02
=N       50
```

A lamp test can be carried out by means of input 05, i. e. if input 05 is switched to "1" while 54 is flashing, the flashing light becomes a steady light.

11.3 New value signalling with double flashing light

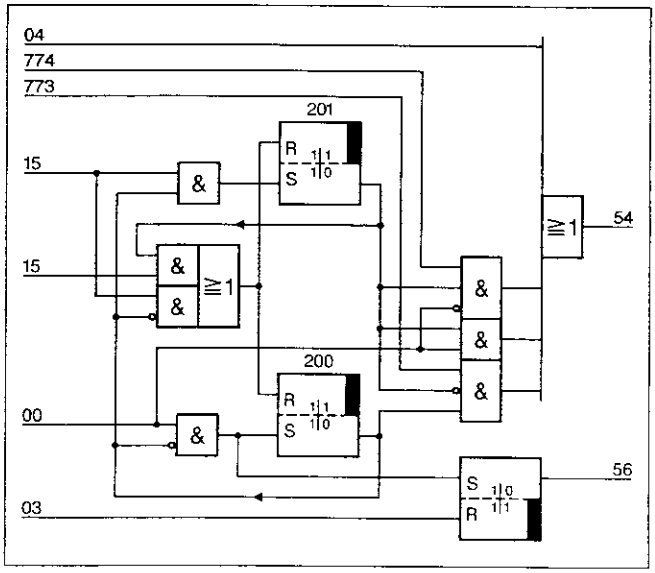


Figure 11.5

00 – Signal input, condition
03 – Acknowledgement, audible signal
04 – Lamp test
10 – Acknowledgement, signal buffer (clear)
15 – Acknowledgement for signal
54 – Indicator lamp output
56 – Audible output
773 – Flashing bus, 5 Hz
774 – Flashing bus, 0.5 Hz

Each new condition at the signal input 00 is indicated by 5 Hz flashing at output 54. At the same time, the audible signal at output 56 is activated. Pression of the acknowledgement key 15 causes the flashing light to switch to a slow blinking light (0.5 Hz) if the condition no longer exists. If the condition still exists, the flashing light switches to continous light. Slow flashing can be cleared with the acknowledgement key 10 for the signal memory. The audible signal at output 56 is cleared with its own acknowledgement key (03).

Function diagram

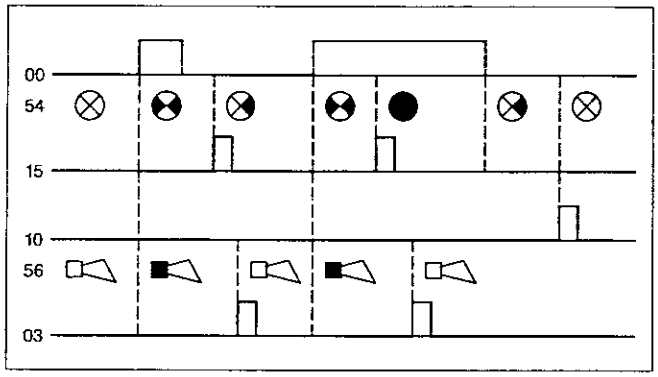


Figure 11.6

Program:

```
!        00
&N      200
FUN      02
=       200
FUN      02
=        56
!        15
&       200
FUN      02
=       201
!        15
&N      200
STR     201
&        10
/       STR
FUN      02
=N      200
FUN      02
=N      201
```

```
!        200
&N       201
&        773
STR      201
&         00
STR      201
&N        00
&        774
/        STR
/        STR
/         04
=         54
!         03
FUN       02
=N        56
```

If a 2 Hz oscillator is required instead of the one shown in the example, then & 211 should be programmed instead of & 773. In addition, the following program section must be added (see also the section on timer functions):

Program:

```
!N       T00
=   T00 0.25
!        T00
FUN   00 210
!N       210
FUN       06
!N       211
=        211
FUN       07
```

11.4 Initial value signalling with single acknowledgement to DIN 19 235

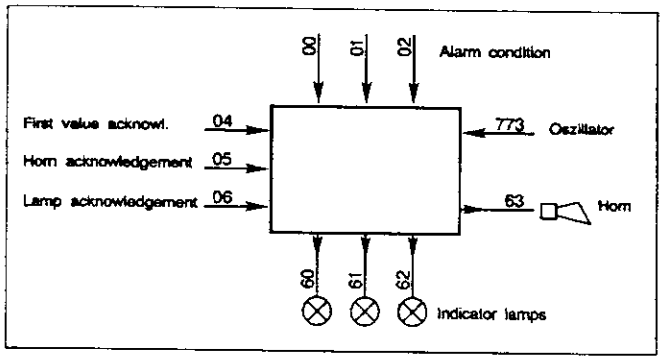


Figure 11.7

The first value signalling circuit emphasizes the alarm condition at inputs 00 – 02 which occurred first within a group. The first alarm condition which occurs is signalled by a flashing light (5 Hz), while all others have a steady light. Acknowledgement is possible only with the first value acknowledgement key 04. If the alarm condition still exists after first value acknowledgement, the lamp switches to continous light until the alarm condition disappears.

All other alarm conditions remain indicated by a continous light until acknowledged and until the alarm condition disappears.
The horn is switched on each time a new alarm condition is signalled.

Function diagram

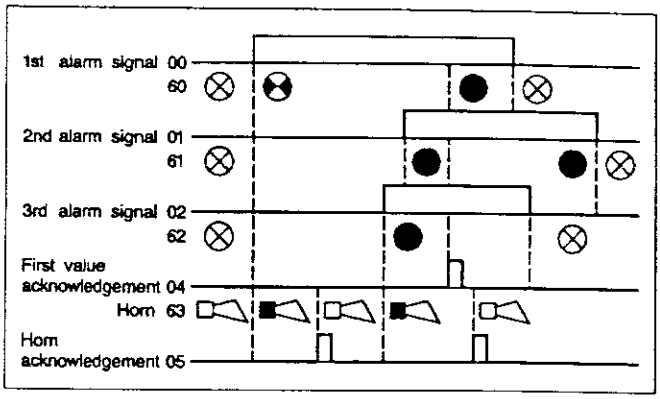


Figure 11.8

Program:

```
!         00
FUN    00 200
!         01
FUN    00 201
!         02
FUN    00 202
!        200      ┐
/        201      │
/        202      │ Horn on
FUN       02      │
=         63      ┘

!         04      ┐
FUN       02      │
=N       210      │ Acknowledgement
=N       211      │
=N       212      ┘
=N       220      ┐
=N       221      │ First value
=N       222      │
=N       223      ┘
!        200
FUN       02
=        210
!        201
FUN       02
=        211
!        202
FUN       02
=        212
```

```
!      773
/      220
&      210
STR     00
&N     210
STR     06
/      STR
/      STR
=       60
!      773
/      221
&      211
STR     01
&N     211      Indicator lamps
STR     06
/      STR
/      STR
=       61
!      773
/      222
&      212
STR     02
&N     212
STR     06
/      STR
/      STR
=       62

!       05
FUN     02      Horn acknowledgement
=N      63

!      210
/      212
&N     223
FUN     02
=      221
!      210
/      211
&N     223
FUN     02
=      222      First value evaluation
!      211
/      212
&N     223
FUN     02
=      220
!      210
/      211
/      212
FUN     02
=      223
```

11.5 Initial and new value signalling with double acknowledgement to DIN 19 235

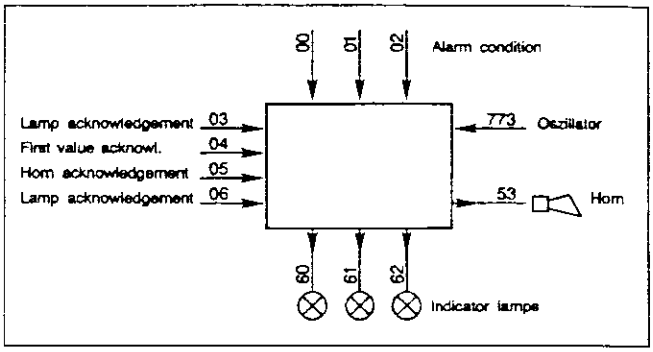


Figure 11.9

In the case of first and new value signalling, all alarm states are first displayed as new values with a flashing light, (5 Hz). After acknowledgement of the lamp at input 03, only the first value signal continues to flash. The others remain displayed with a continous light as long as the alarm condition exists. The first value can be acknowledged only with the first value acknowledgement key 04, and remains displayed as a steady light until the condition disappears.

Function diagram

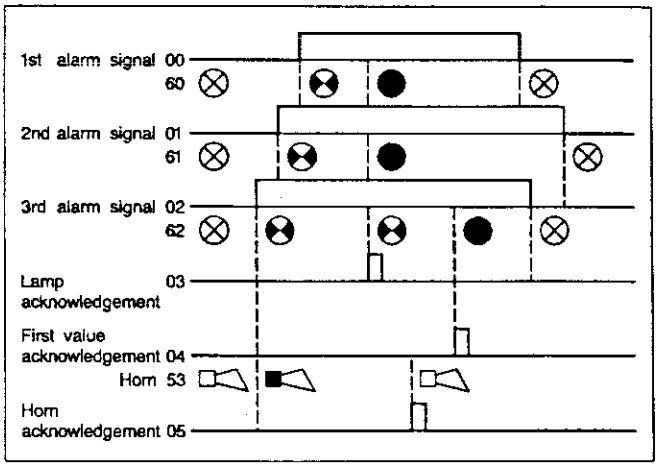


Figure 11.10

Program:

```
!          00
FUN    00 200
!          01
FUN    00 201
!          02
FUN    00 202
!         200
/         201
/         202
FUN        02
=          53
```

```
!      03
&      220
FUN    02
=N     210
!      03
&      221
FUN    02
=N     211
!      03
&      222
FUN    02
=N     212

!      200
FUN    02
=      210
!      201
FUN    02
=      211
!      202
FUN    02
=      212

!      773
&      210
STR    00
&N     210
STR    06
/      STR
/      STR
=      60
!      773
&      211
STR    01
&N     211
STR    06
/      STR
/      STR
=      61
!      773
&      212
STR    02
&N     212
STR    06
/      STR
/      STR
=      62

!      05
FUN    02
=N     53
```

Lamp acknowledgement

Lamps

Horn acknowledgement

```
!      210
/      212
&N     223
FUN    02
=      221
!      210
/      211
&N     223
FUN    02
=      222
!      211
/      212
&N     223
FUN    02
=      220
!      210
/      211
/      212
FUN    02
=      223

!      04
FUN    02
=N     220
=N     221
=N     222
=N     223
!      03
FUN    02
=      204
!      04
&      204
FUN    02
=N     210
=N     211
=N     212
=N     204
```

First value evaluation

First value acknowledgement

Notes

12.1 Function 30 (load word) Function 32 (word output)

Program:

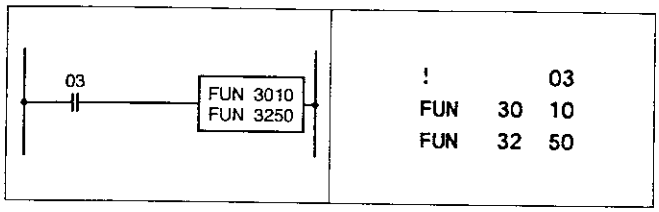


Figure 12.1

A "0–1" edge at input 03 causes the status of inputs 10 to 27 to be transferred to the arithmetic register (with FUN 30) and then sent to outputs 50 to 67 with FUN 32. When input 03 becomes inactive ("0" signal), the outputs remain set until a "1" signal at input STOP interrupts program execution or a voltage failure resets the system. It is also possible for new information to be entered at inputs 10 to 27 with input 03. If input 03 is continuously active, any changes to the inputs are immediately sent to the outputs.

12.2 Function 31 (load word from T/Z) Function 32 (word output)

Program

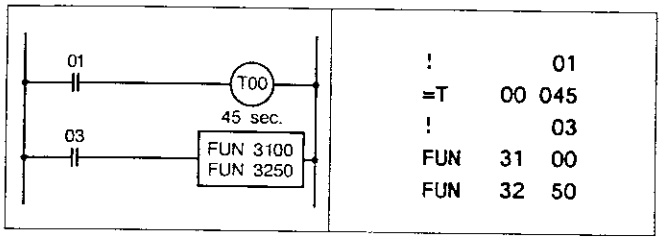


Figure 12.2

A "0–1" edge at input 03 transfers the value of the timer T00 to the arithmetic register (with FUN 31) and FUN 32 transfers this information to outputs 50 to 67 (stored).

If input 03 is continously active ("1" signal), the current time value is immediately transferred to the outputs (4 decades BCD).

12.3 Function 30 (load word) Function 33 (load T/Z)

Program:

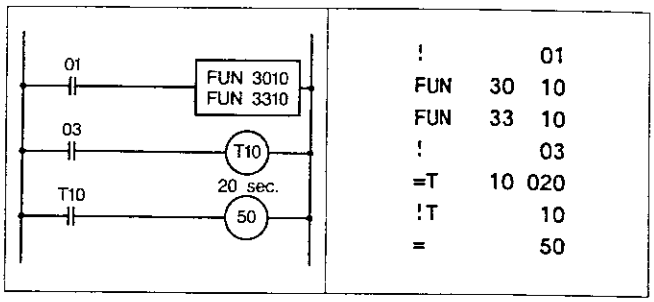


Figure 12.3

A "0–1" edge at input 01 transfers the status of inputs 10 to 27 to the arithmetic register (with FUN 30).
FUN 33 transfers the contents of the arithmetic value as the actual value to timer 10. The timer T10 cannot be started as long as the pulse at input 01 is present.
After the time has elapsed, output 50 is set ("1" signal).
At the start of the cycle, the programmed value of the timer is valid (without a pulse at input 01).

Caution: The information at the inputs must be in BCD.

12.4 Function 36 (load contents of high speed counter), Function 32 (word output)

Program:

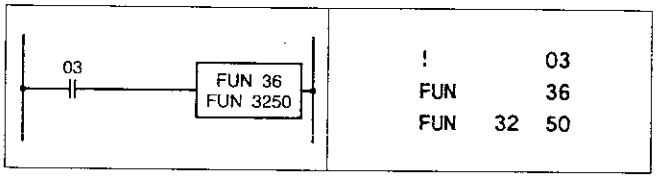


Figure 12.4

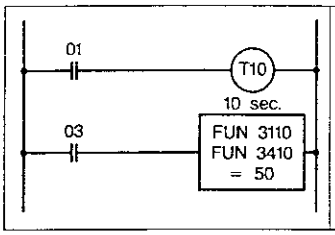
A "0–1" edge at input 03 transfers the current value of the high speed counter to the arithmetic register (with FUN 36). FUN 32 transfers this value to outputs 50 to 67 (stored).

If input 03 is continously active, the current counter value is continously displayed at the outputs.

The fast counter is an up counter (10 kHz counting frequency, 4 decades) with the external counter input HZ and the reset input HR.

12.5 Function 31 (load from T/Z), Function 34 (compare word for less than or equal to)

Program




```
!          01
=T    10 010
!          03
FUN   31   10
FUN   34   10
=          50
```

Figure 12.5

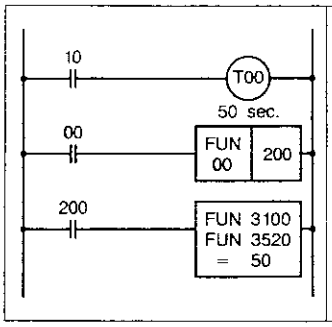
An "0–1" edge at input 03 transfers the value of the timer T10 to the arithmetic register (with FUN 31) and FUN 34 compares this with the status of inputs 10 to 27. If the value of inputs 10 to 27 is greater than or equal to the value of the timer, then output 50 is activated ("1" signal). If the value of the inputs is less than the value of the timer, output 50 remains off ("0" signal).

If input 03 is continously active ("1" signal), continous comparison between inputs and the timers is carried out after the time has elapsed.

When input 3 is turned off ("0" signal) the status is filed at output 50.

12.6 Function 31 (Load from T/Z), Function 34 (Compare with T/Z)

Program:




```
!          10
=T    00 050
!          00
FUN   00 200
!          200
FUN   31   00
FUN   35   20
=          50
```

Figure 12.6

When input 10 carries a "1" signal, the timing element T00 is started.

When a "1" signal is put on input 00, an impulse is formed at marker 200 by FUN 00; its length equals the cycle time.

With this impulse the actual value of the timing element T00 is then transferred to the arithmetic register. The actual value of the timing element T20 is compared with its setpoint value by FUN 35.

If the value of the timing element T00 is smaller than or equal to T20, output 50 is set ("1" signal). If the value of T00 is larger than T20, output 50 remains in resting state ("0" signal).

13.1 Special functions (770 - 777)

Special functions are stored in the form of software and addresses 770 to 777. They are also programmed and called with these numbers.

No. 770 Outputs all OFF (assignment)

When output no. 770 is turned on with a program, all outputs are automatically turned OFF. This function allows to control inputs or outputs (flags), which mustn't switched on at the same time.

Example:

If the input 0 and input 1 have "1" signal at the same time, all outputs are reset, and the arithmetical lapse is interrupted. In the display the message 770E appears.

If the error is corrected, press the key STO and then the key STA or switch the power supply off and then on and press after that the key STA. Then the user program will be continued.

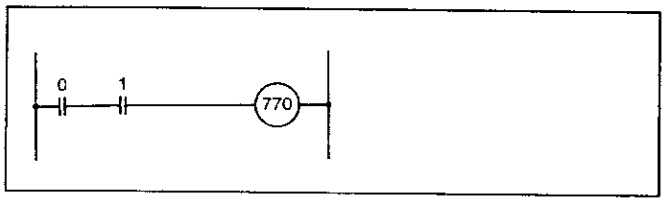


Figure 13.1

No. 771 Resetting of all buffered values such as flags, timers, counters. This function is used at the beginning of a program to reset all buffered flags, timers, and counters, e. g. after power failure or switch off, if states before power failure or switch off shall not be considered. As the function 771 is static, i. e. it is executed once per program cycle, it is appropriate to use No. 771 combined with No. 777 (one pulse after program start, s. figure 13.2).

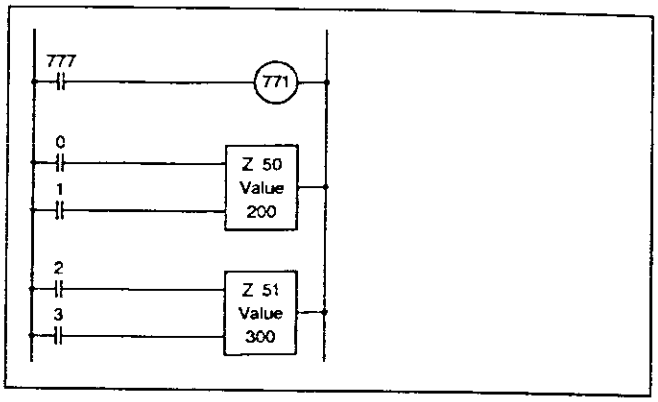


Figure 13.2

No. 772 Cycle time oscillator (1 cycle pulse, 1 cycle pause)

No. 773 0.1 s oscillator

No. 774 1 s oscillator

No. 775 0.01 s oscillator

Caution: Can be used only when the cycle time is less than 0.01 s.

No. 776 1 minute oscillator

No. 777 One pulse after program start (cycle time, s. figure 13.3)

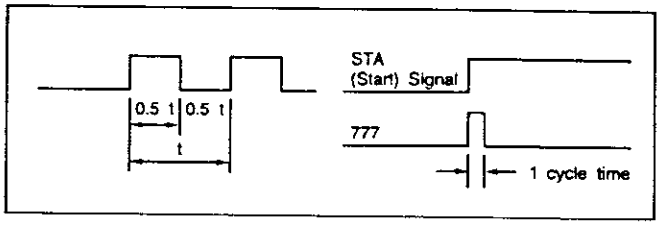


Figure 13.3

The duty cycle of each oscillator output is 1:1 (s. figure 13.4).

Example: No. 774 (0.5 s pulse, 0.5 s pause)
No. 772 (1 cycle pulse, 1 cycle pause)

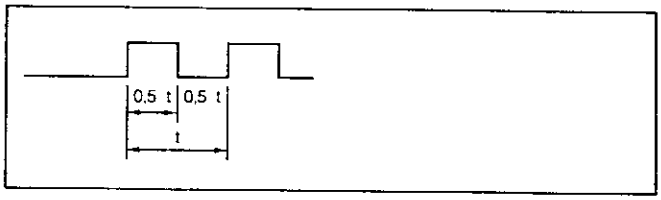


Figure 13.4

14.1 Programming concept

Please note the following points:

a. The number of contacts for inputs/outputs, flags, timers and counters is not restricted; any number of contacts can be assigned (Figure 14.1).

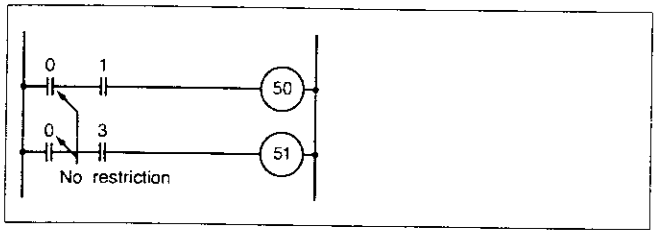


Figure 14.1

b. There is no restriction to the number of contacts which can be connected in series or parallel (Figure 14.2).

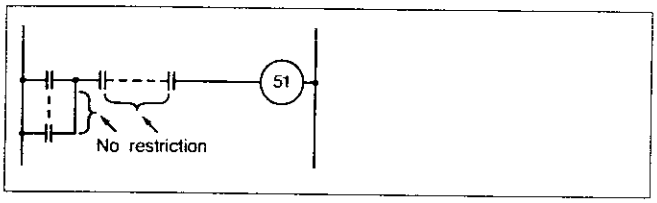


Figure 14.2

c. An output (including timer/counter) cannot be assigned without an input condition (Figure 14.3 upper part: not allowed assignment, figure 14.3 lower part: allowed assignment).

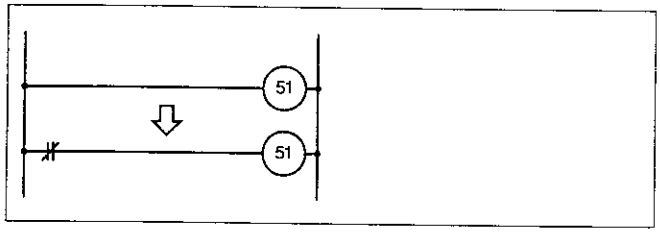


Figure 14.3

d. It is not permitted to append a further logical operation after an output assignment (figure 14.4 upper part).
Ensure that the element to be processed was programmed before the output assignment (Figure 14.4 lower part).

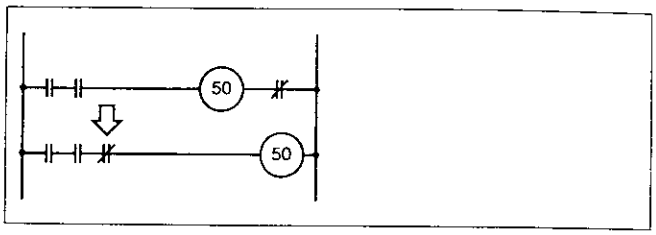


Figure 14.4

e. An output (timer/counter) can be assigned only once in the program (see upper part of figure 14.5). If this output is required for several circuit constellations, then the circuit diagram (program) must be modified as shown in the lower part of figure 14.5.

Note: This is not true for FUN 02, FUN 03, FUN 06 and FUN 07.

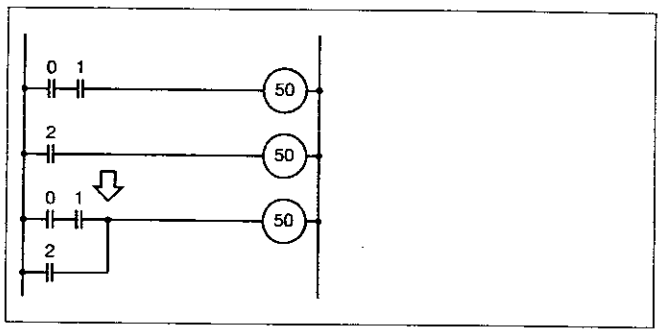


Figure 14.5

f. Ensure that programming is carried out from left to right (see Figure 14.1 to 14.5).

g. When programming counters or shift registers which have two or more inputs, specific (defined) programming commands must be used (e.g. Figure 14.6).

h. During programming, timer or counter values are entered immediately with the T/Z assignment (Figure 14.7).

i. A step number (STEP No.) or an input/output is programmed with a 3-digit number. However, leading zeros can be omitted.

Examples: – Input 5 can be entered as 5, 05 or 005. In the display, only input 5 appears.

– Step number 050 can be entered as step 50 or 050.

j. For programmed times shorter than 99 seconds, it is possible, for example, to program 099 or 99,0 or 99. The display always shows 99,0.

Example:	Input	Display
	010	10.0
	15	15.0
	020	20.0

Program:

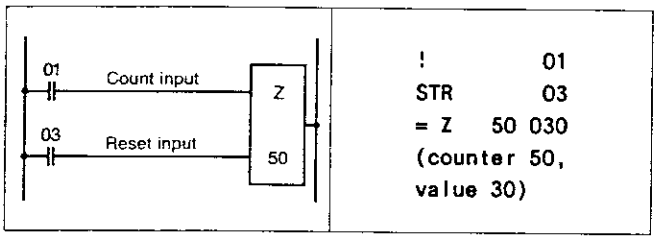


Figure 14.6

Program:

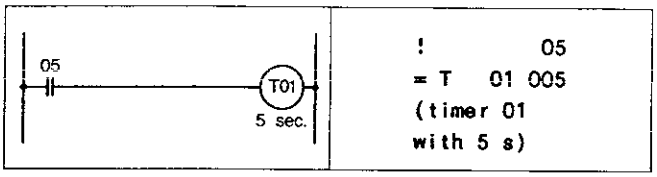


Figure 14.7

14.2 Recommended programming procedure

a. Use the ladder diagram as an aid for program preparation (example figure 14.8).

b. Subdivide the ladder diagram into blocks, beginning at the top left with the starting conditions. In figure 14.9 blocks (1) to (11) are shown, as they have to be subdivided in the given example.

c. Combine the individual blocks in the order stated in figure 14.10, 3rd column until you have made your way towards the right-hand side and reached the output assignment.

Example

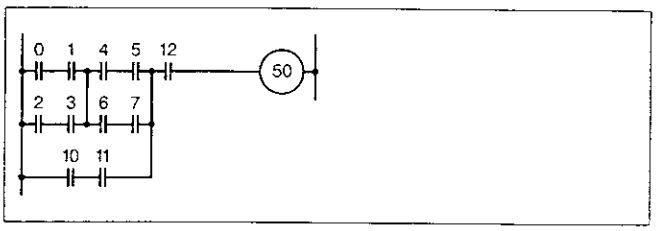


Figure 14.8

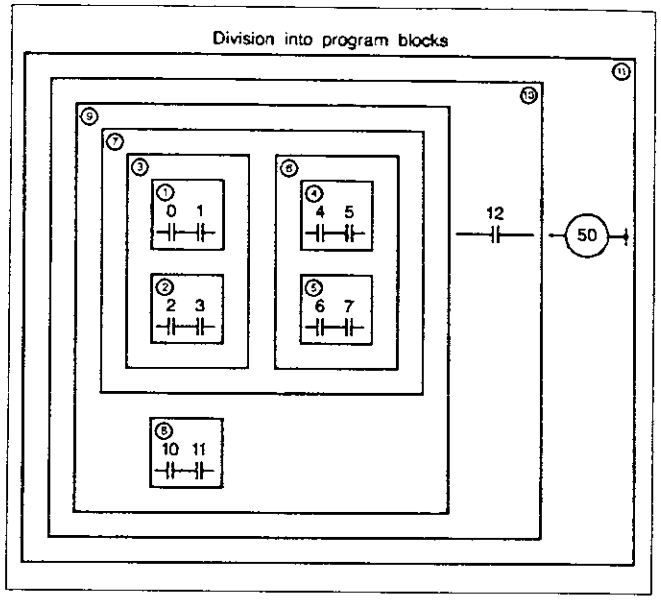


Figure 14.9

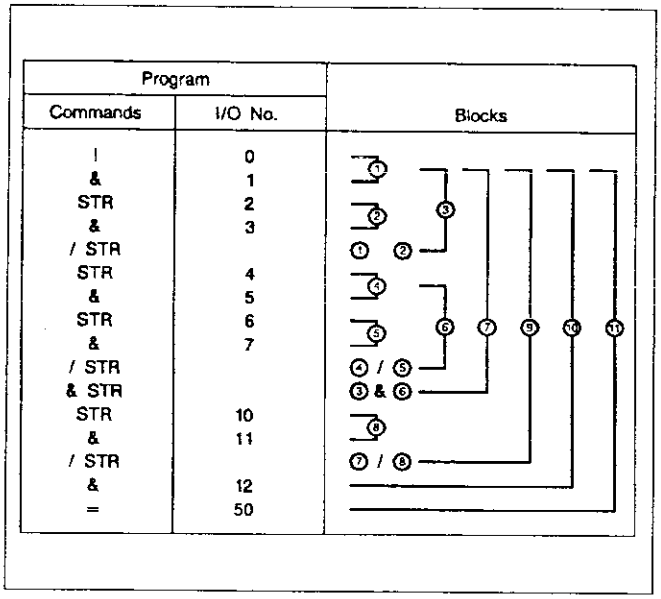


Figure 14.10

14.3 User program memory

a. 950 statements can be programmed with the program memory in the system (EEPROM). 1970 statements can be programmed with the external memory module 07 PR 201 (EEPROM) or with the external memory module 07 PR 210 (EPROM) resp. The step number starts with decima 000.

b. Creation of each new user program always starts at step 000.

c. Program sections which are added always start at the first free address of the user program memory (see figure 14.11).

d. An invalid program step cannot be transferred into a user program (error message).

e. If addresses are inserted into or deleted from the program memory, the address numbers of all other statements are shifted automatically either up or down (see figures 14.12 and 14.13).

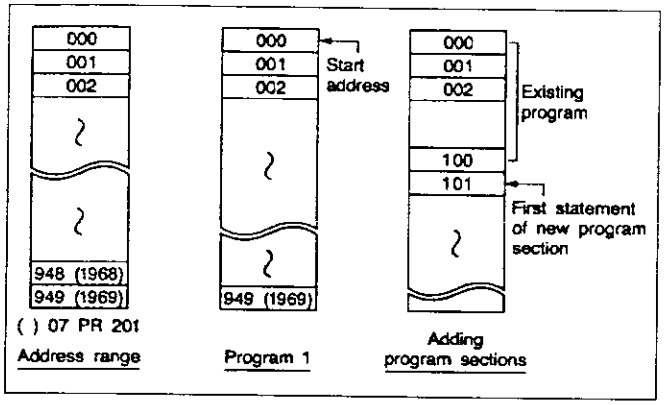


Figure 14.11

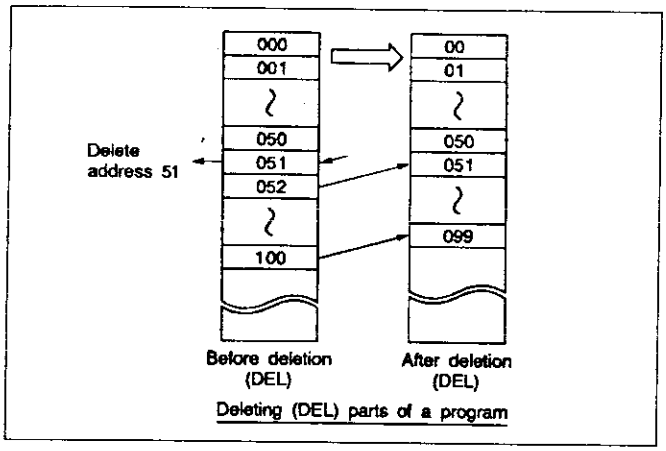


Figure 14.12

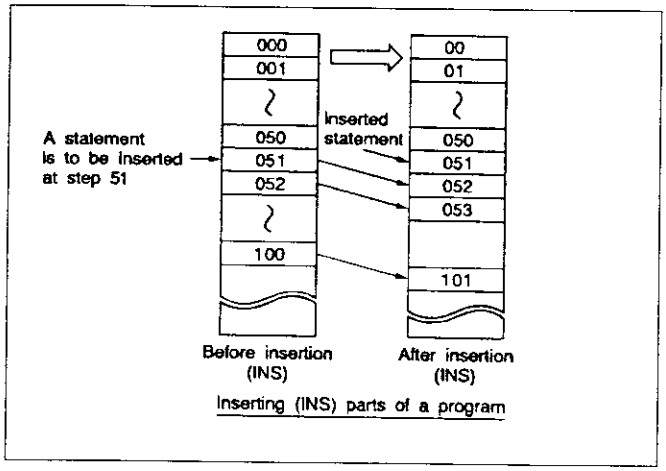


Figure 14.13

14.4 Application examples

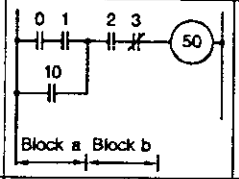
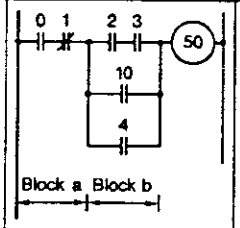
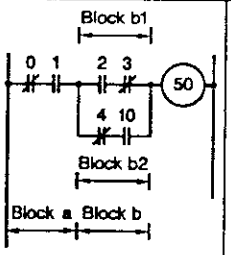
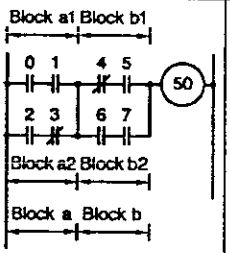
	Circuit	Program Comm.		Remarks
Parallel/series connection		! & / & & N =	0 1 10 2 3 50	a b
Series/parallel connection		! & N STR & / / & STR =	0 1 2 3 10 4 50	a b Two blocks a and b are combined with & STR
		! N & STR & N STR N & / STR & STR =	0 1 2 3 4 10 50	a b1 b2 Combine b1 and b2 with / STR Two blocks a and b are combined with & STR
Two parallel circuits in series		! & STR & N / STR STR N & STR & / STR & STR =	0 1 2 3 4 5 6 7 50	a1 a2 Parts a1 and a2 are combined with / STR b1 b2 Combine b1+b2 with /STR Combine blocks a+b with & STR

Figure 14.14

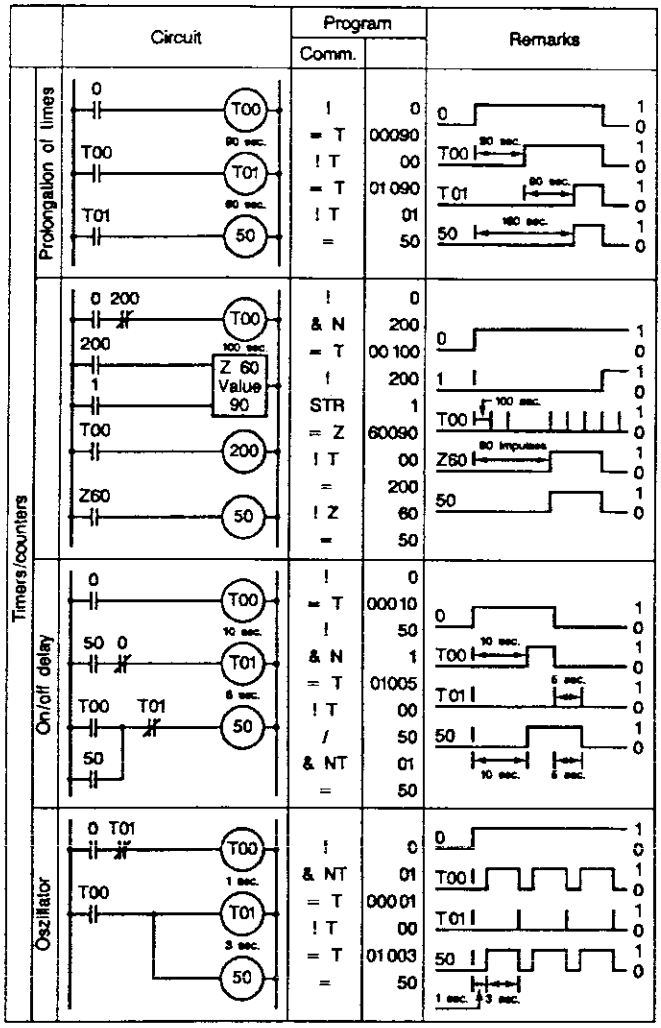


Figure 14.15

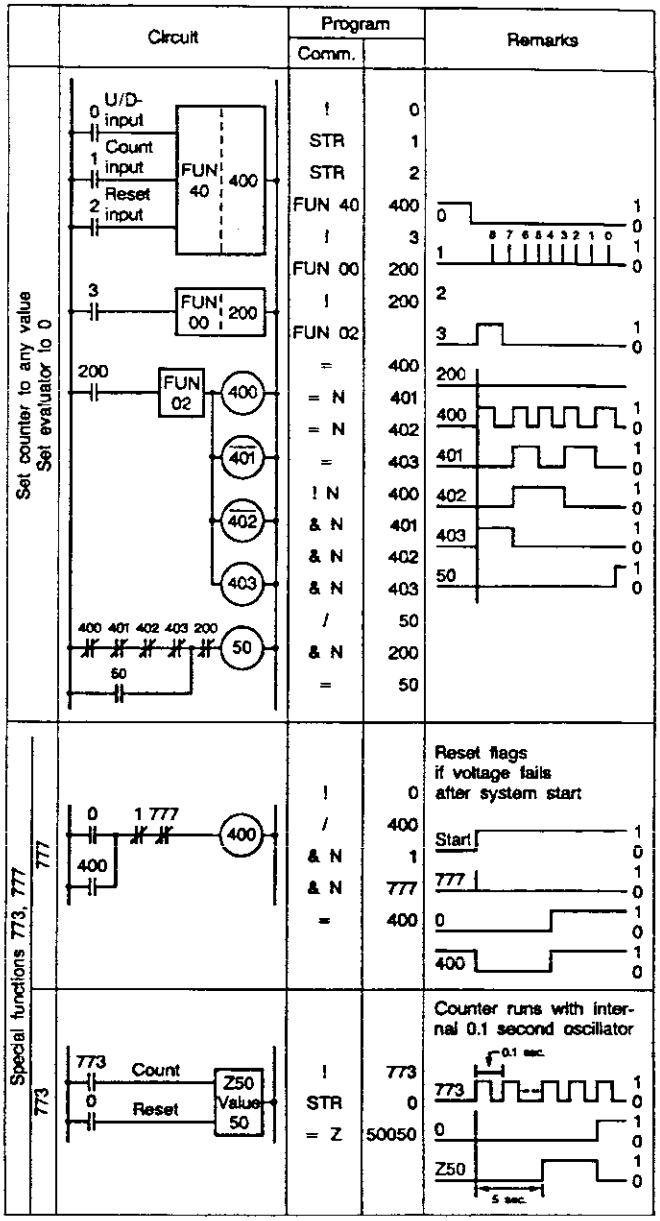


Figure 14.16

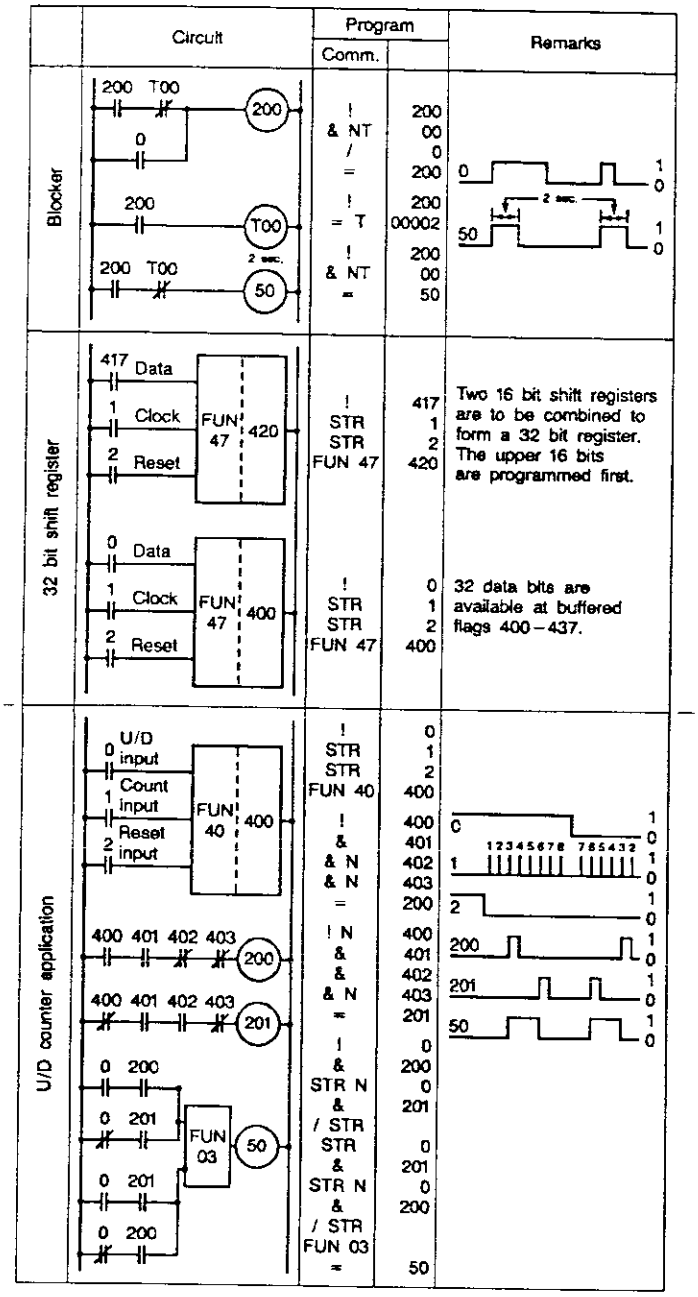


Figure 14.17

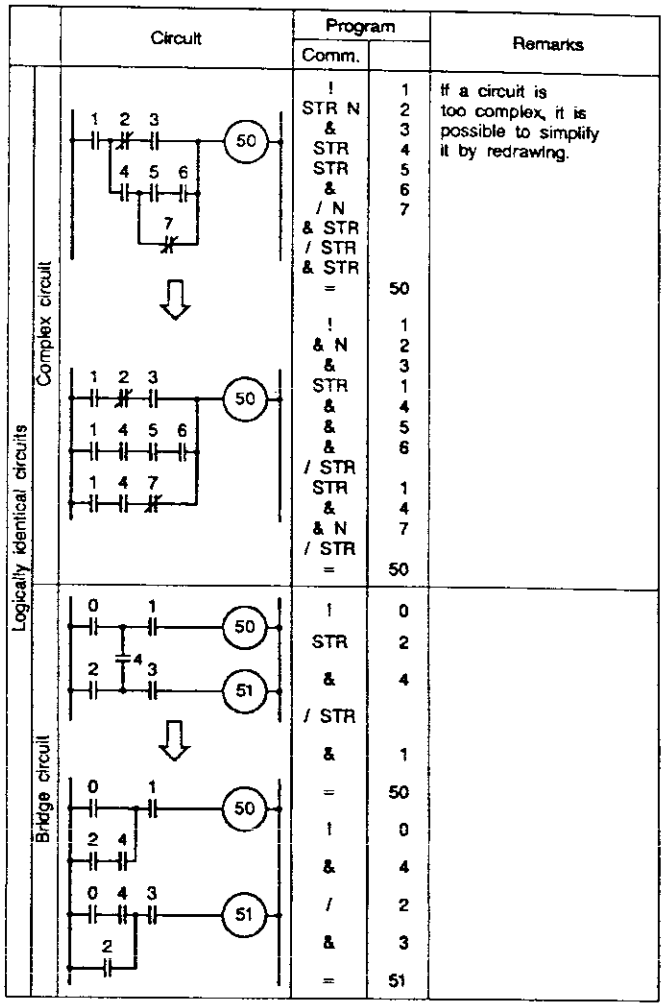


Figure 14.18

Notes

15 Forms for project planning

This section includes the following forms for project planning of the PROCONTIC K200:

Operand list – inputs

Operand list – outputs

Variable list (miscellaneous)

Variable list (unbuffered flags)

Variable list (buffered flags)

Instructions list

Input		Input		Input		Input	
00		40		100		140	
01		41		101		141	
02		42		102		142	
03		43		103		143	
04		44		104		144	
05		45		105		145	
06		46		106		146	
07		47		107		147	
10				110		150	
11				111		151	
12				112		152	
13				113		153	
14				114		154	
15				115		155	
16				116		156	
17				117		157	
20				120		160	
21				121		161	
22				122		162	
23				123		163	
24				124		164	
25				125		165	
26				126		166	
27				127		167	
30				130		170	
31				131		171	
32				132		172	
33				133		173	
34				134		174	
35				135		175	
36				136		176	
37				137		177	

Inputs 00 ... 47
100 ... 147 with EA 264, 100 ... 177 with 07 EB 200, 07 EB 205

PROCONTIC K 200	Operand list – inputs				
Program name		Programmed			Sheet
		Checked before loading			
Program title		Checked after loading			
		Commissioned			of
Program for		Order No.			

50	51	52	53	54	55	56	57	60	61	62	63	64	65	66	67

70	71	72	73	74	75	76	77

100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137

140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177

Outputs 50 ... 77
150 ... 177 with EA 264, 100 ... 177 with 07 AB 200, 07 AB 205

PROCONTIC K200 | Operand list – outputs

Program name

Program title

Program for

Programmed			Sheet
Checked before loading			
Checked after loading			
Commissioned			of

Order No.

Program for

Program title

Program name

PROCONTIC K 200

Order No.

Commissioned | Checked after loading | Checked before loading | Programmed

of | Sheet

miscellaneous variables

Timers T		Timers T		Counters Z		Special functions	
Ø0		40		50		770	Reset all outputs to zero
01		41		51		771	Reset data after voltage failure
02		42		52		772	Cycle time oscillator
03		43		53		773	0,1 second oscillator
04		44		54		774	1 second oscillator
05		45		55		775	0,01 second oscillator
06		46		56		776	1 minute oscillator
07		47		57		777	1 pulse ater STA
10				60			
11				61			
12				62			
13				63			
14				64			
15				65			
16				66			
17				67			
20				70			
21				71			
22				72			
23				73			
24				74			
25				75			
26				76			
27				77			
30							
31							
32							
33							
34							
35							
36							
37							

Timers (0-1 delay) T 00 ... T 47 | Down counters Z 50 ... Z 77 | Special functions 770 ... 777

200		240		300		340	
201		241		301		341	
202		242		302		342	
203		243		303		343	
204		244		304		344	
205		245		305		345	
206		246		306		346	
207		247		307		347	
210		250		310		350	
211		251		311		351	
212		252		312		352	
213		253		313		353	
214		254		314		354	
215		255		315		355	
216		256		316		356	
217		257		317		357	
220		260		320		360	
221		261		321		361	
222		262		322		362	
223		263		323		363	
224		264		324		364	
225		265		325		365	
226		266		326		366	
227		267		327		367	
230		270		330		370	
231		271		331		371	
232		272		332		372	
233		273		333		373	
234		274		334		374	
235		275		335		375	
236		276		336		376	
237		277		337		377	

Unbuffered flags 200 ... 377

PROCONTIC K200

Variable list
Unbuffered flags

Program name

Program title

Program for

Programmed

Checked before loading

Checked after loading

Commissioned

Sheet

of

Order No.

400	401	402	403	404	405	406	407	410	411	412	413	414	415	416	417	420	421	422	423	424	425	426	427	430	431	432	433	434	435	436	437
440	441	442	443	444	445	446	447	450	451	452	453	454	455	456	457	460	461	462	463	464	465	466	467	470	471	472	473	474	475	476	477
500	501	502	503	504	505	506	507	510	511	512	513	514	515	516	517	520	521	522	523	524	525	526	527	530	531	532	533	534	535	536	537
540	541	542	543	544	545	546	547	550	551	552	553	554	555	556	557	560	561	562	563	564	565	566	567	570	571	572	573	574	575	576	577

Buffered flags 400 ... 577

PROCONTIC K200 | Variable list Buffered flags

Program name

Program title

Program for

Programmed			Sheet
Checked before loading			
Checked after loading			
Commissioned			of

Order No.

600	601	602	603	604	605	606	607	610	611	612	613	614	615	616	617	620	621	622	623	624	625	626	627	630	631	632	633	634	635	636	637
640	641	642	643	644	645	646	647	650	651	652	653	654	655	656	657	660	661	662	663	664	665	666	667	670	671	672	673	674	675	676	677
700	701	702	703	704	705	706	707	710	711	712	713	714	715	716	717	720	721	722	723	724	725	726	727	730	731	732	733	734	735	736	737
740	741	742	743	744	745	746	747	750	751	752	753	754	755	756	757	760	761	762	763	764	765	766	767	770	771	772	773	774	775	776	777

Buffered flags 600 ... 767

PROCONTIC K 200 | Variable list Buffered flags

Program name

Program title

Program for

Programmed | Checked before loading | Checked after loading | Commissioned

Sheet

of

Order No.

Step Nr. Wortadr.	Statement	Symbolic name
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

Commands	Name
!	If
!N	If not
STR	Store
STR N	Store not
&	And
&N	And not
/	Or
/N	Or not
& STR	And store
/STR	Or store
=	Then
= N	Then not
FUN 0	Load constant
FUN 00	Pulse generator
FUN 02	Set/reset output/flag
FUN 03	RS memory
FUN 04	Set MC
FUN 05	Reset MC
FUN 05	Jump to FUN 07
FUN 07	End of jump
FUN 7	Register ≧ constant
FUN 8	Register = constant
FUN 9	Register < constant
FUN 11	Addition
FUN 12	Subtraction
FUN 13	Multiplication
FUN 14	Division
FUN 22	Output word
FUN 24	Bin → BCD
FUN 25	BCD → Bin
FUN 30	Load word I/Os
FUN 31	Load word from T/Z
FUN 32	Load word to output A/flag M
FUN 33	Load word to T/Z
FUN 34	Compare I/Os with register
FUN 35	Compare register with T/Z
FUN 36	Load contents of high speed counter
FUN 40	U/D counter
FUN 45	Dynamic memory
FUN 47	Shift register
FUN 99	End
T/Z	Timer/Counter

Address assignments:

Inputs	00-47, 100-147
Outputs	50-77, 150-177
Flags	200-377
Flags, buffered	400-767
Special functions	770-777
Timers	T00-T47
Counters	Z50-Z77

(Octal addressing permitted only for digits O-7)

PROCONTIC K200	Instructions list			
Program name	Programmed			Sheet
	Checked before loading			
Program title	Checked after loading			
	Commissioned			of
Program for	Order No.			

Commands	Name
!	If
!N	If not
STR	Store
STR N	Store not
&	And
&N	And not
/	Or
/N	Or not
& STR	And store
/STR	Or store
=	Then
= N	Then not
FUN 0	Load constant
FUN 00	Pulse generator
FUN 02	Set/reset output/flag
FUN 03	RS memory
FUN 04	Set MC
FUN 05	Reset MC
FUN 05	Jump to FUN 07
FUN 07	End of jump
FUN 7	Register ≧ constant
FUN 8	Register = constant
FUN 9	Register < constant
FUN 11	Addition
FUN 12	Subtraction
FUN 13	Multiplication
FUN 14	Division
FUN 22	Output word
FUN 24	Bin → BCD
FUN 25	BCD → Bin
FUN 30	Load word I/Os
FUN 31	Load word from T/Z
FUN 32	Load word to output A/flag M
FUN 33	Load word to T/Z
FUN 34	Compare I/Os with register
FUN 35	Compare register with T/Z
FUN 36	Load contents of high speed counter
FUN 40	U/D counter
FUN 45	Dynamic memory
FUN 47	Shift register
FUN 99	End
T/Z	Timer/Counter

Address assignments:

Inputs	00-47, 100-147
Outputs	50-77, 150-177
Flags	200-377
Flags, buffered	400-767
Special functions	770-777
Timers	T00-T47
Counters	Z50-Z77

(Octal addressing permitted only for digits 0-7)

PROCONTIC K200

Program name	Programmed			Sheet
	Checked before loading			
Program title	Checked after loading			
	Commissioned			of
Program for	Order No.			

Notes

Printed on chlorine-free bleached paper

ABB Schalt- und Steuerungstechnik GmbH
Eppelheimer Straße 82 Postfach 10 50 09
D-69123 Heidelberg D-69040 Heidelberg

Telephone +49 6221 777-0
Telefax +49 6221 777-111

Printed in the Federal Republic of Germany (03.94 · 0.06 · AD)