Industrial Automation

(Automação de Processos Industriais)

PLCs Programming Languages

Instruction List

http://www.isr.ist.utl.pt/~pjcro/courses/api1011/api1011.html

Prof. Paulo Jorge Oliveira
pjcro @ isr.ist.utl.pt
Tel: 21 8418053 ou 2053 (internal)

Syllabus:

Chap. 2 – Introduction to PLCs [2 weeks]

...

Chap. 3 – PLCs Programming Languages [2 weeks]

Standard languages (IEC-1131-3):

Ladder Diagram; Instruction List, and *Structured Text.*

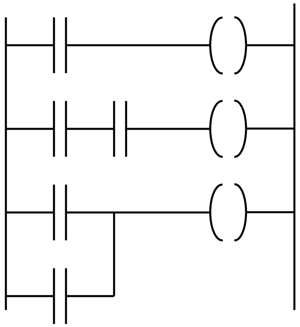
Software development resources.

...

Chap. 4 - GRAFCET *(Sequential Function Chart)* [1 week]

PLCs Programming Languages (IEC 1131-3)

Ladder Diagram



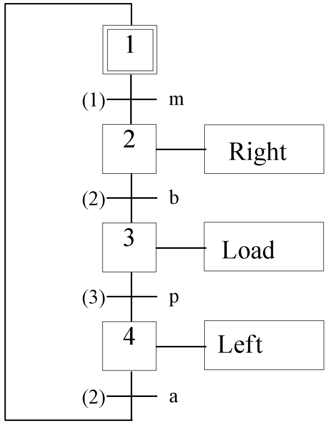
Structured Text

```
If  %I1.0  THEN
  %Q2.1 := TRUE
ELSE
  %Q2.2 := FALSE
END_IF
```

Instruction List

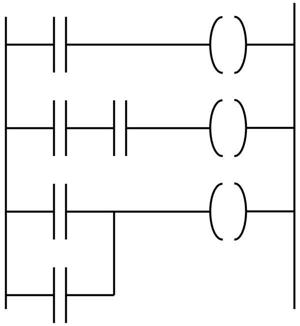
```
LD        %M12
AND       %I1.0
ANDN      %I1.1
OR        %M10
ST        %Q2.0
```

Sequential Function Chart (GRAFCET)



Linguagens de programação de PLCs (IEC 1131-3)

Ladder Diagram



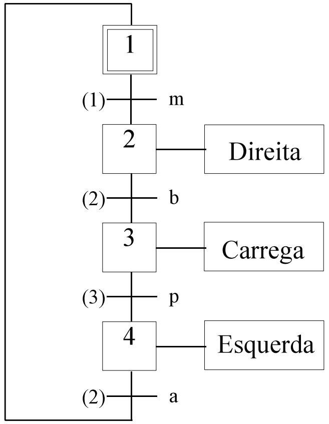
Structured Text

```
If  %I1.0  THEN
  %Q2.1 := TRUE
ELSE
  %Q2.2 := FALSE
END_IF
```

Instruction List

```
LD          %M12
AND         %I1.0
ANDN        %I1.1
OR          %M10
ST          %Q2.0
```

Sequential Function Chart (GRAFCET)



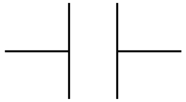
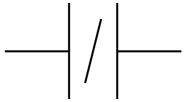
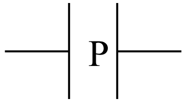
Instruction list

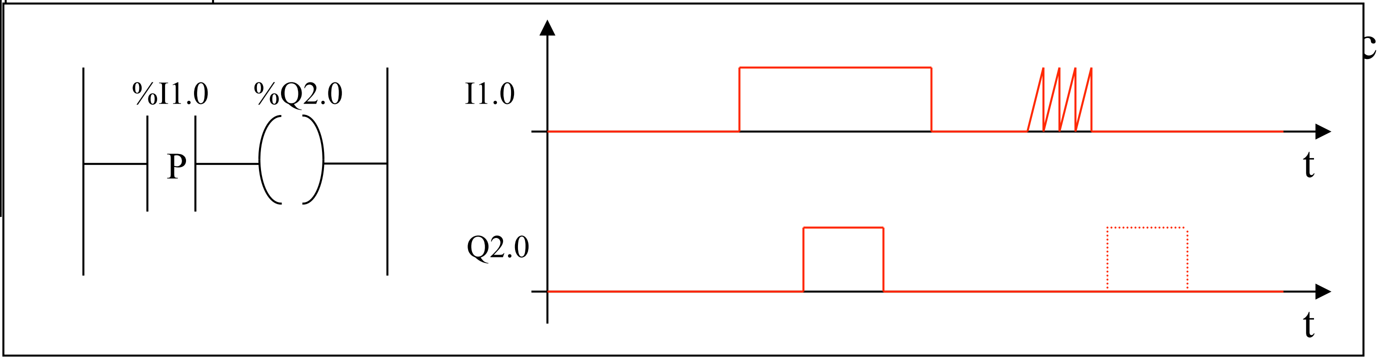
```
ANI1      AI3     LDV50
A(        =P9     =CSW9
OI2       NO      PE
O(        OM1
ANC9      OI4
AQ9       =Z9
)         NO
)         AC9
=Q9       =M1
...       ...
```

Instruction list

Basic Instructions

Load

LD		Open contact: contact is active (result is 1) while the control bit is 1.
LDN		Close contact: contacto is active (result is 1) while the control bit is 0.
LDR		Contact in the rising edge: contact is active during a scan cycle where the control bit has a rising edge.
LDF		cycle

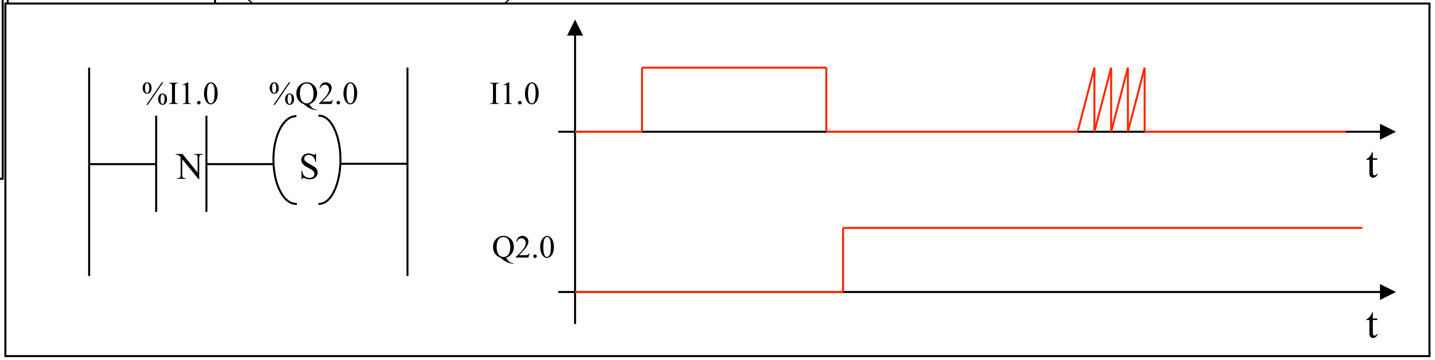


Instruction list

Basic Instructions

Store

ST	—( )—	The result of the logic function activates the coil.
STN	—(/)—	The inverse result of the logic function activates the coil.
S	—(S)—	The result of the logic function energizes the relay (sets the latch).
R		



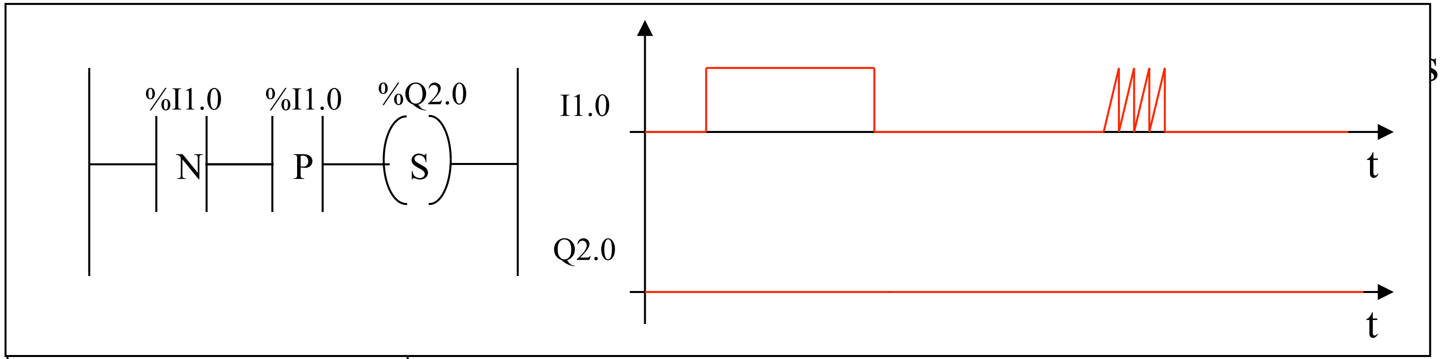
Instruction list

Basic Instructions

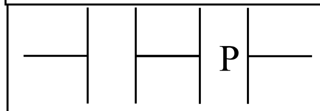
AND

AND

ANDN

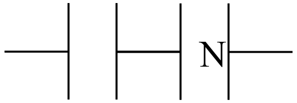


ANDR



AND of the rising edge with the result of the previous logical operation.

ANDF

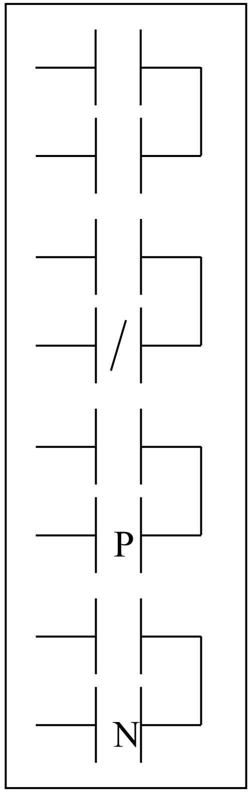


AND of the falling edge with the result of the previous logical operation.

Instruction list

Basic Instructions

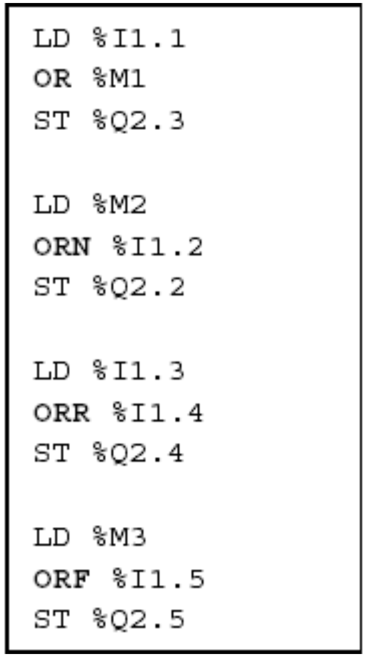
OR



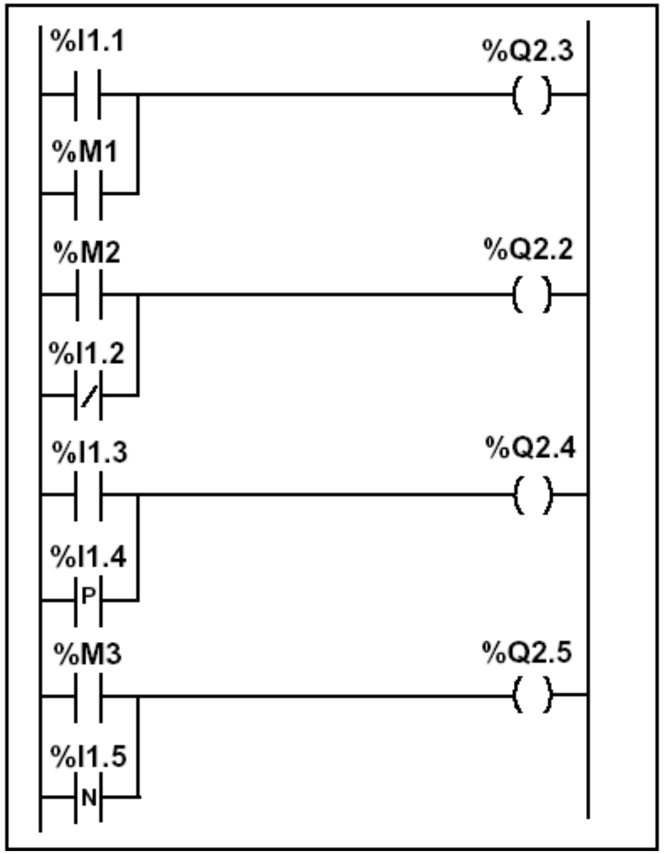
OR	OR of the operand with the result of the previous logical operation.
ORN	OR of the operand with the inverted result of the previous logical operation.
ORR	OR of the rising edge with the result of the previous logical operation.
ORF	OR of the falling edge with the result of the previous logical operation.

Instruction list

Example:



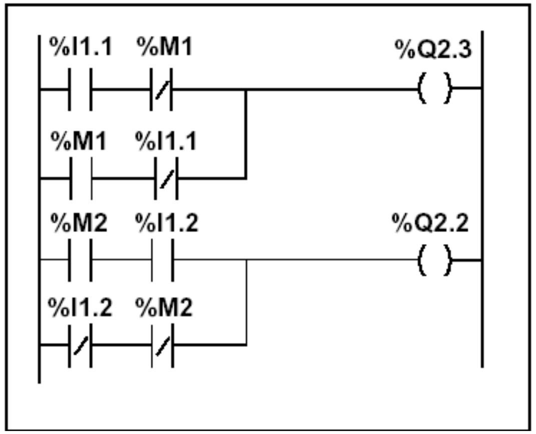
```
LD %I1.1
OR %M1
ST %Q2.3

LD %M2
ORN %I1.2
ST %Q2.2

LD %I1.3
ORR %I1.4
ST %Q2.4

LD %M3
ORF %I1.5
ST %Q2.5
```



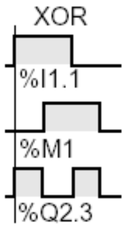
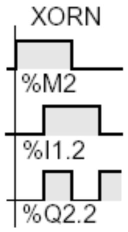
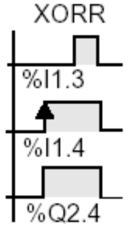
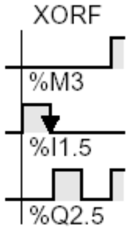
Instruction list

Basic Instructions

XOR



```
...
LD      %I1.1
XOR     %M1
ST      %Q2.3
LD      %M2
XOR     %I1.2
ST      %Q2.2
...
```

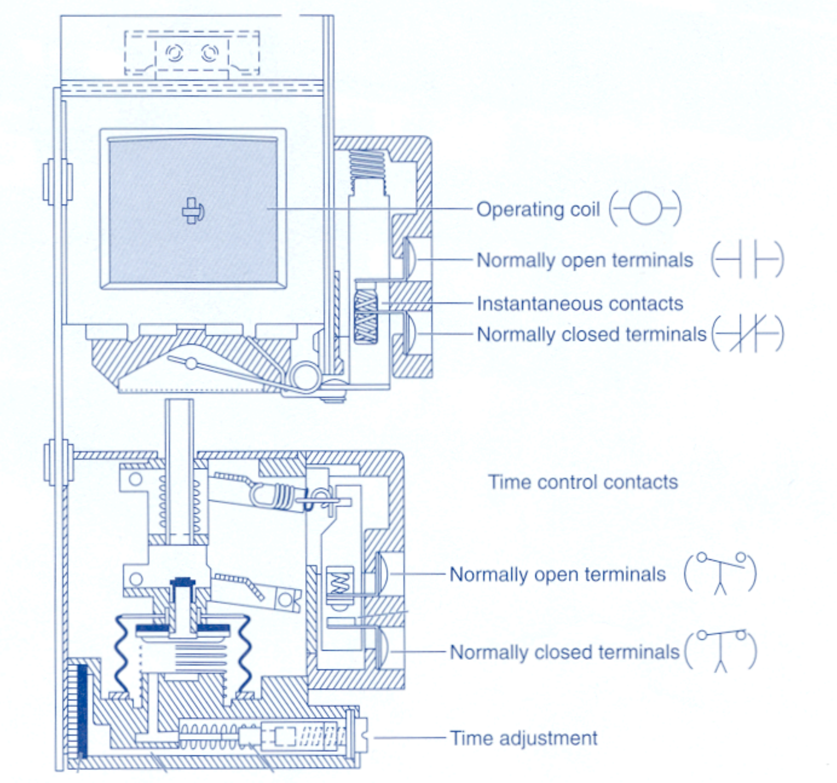
Instruction list	Structured text	Description	Timing diagram
XOR	XOR	OR Exclusive between the operand and the previous instruction's Boolean result	
XORN	XOR (NOT...)	OR Exclusive between the operand inverse and the previous instruction's Boolean result	
XORR	XOR (RE...)	OR Exclusive between the operand's rising edge and the previous instruction's Boolean result	
XORF	XOR (FE...)	OR Exclusive between the operand's falling edge and the previous instruction's Boolean result.	

Instruction list

Temporized Relays

or

Timers



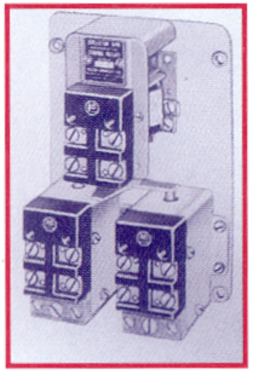


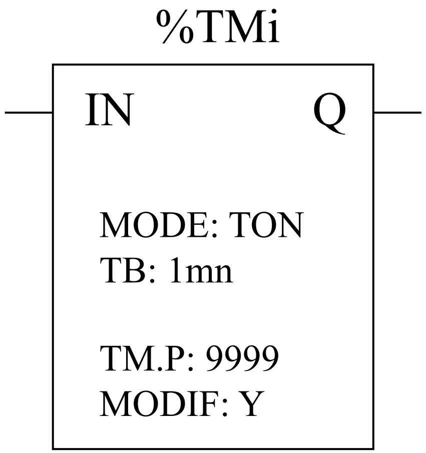
Fig. 7-1
Pneumatic on-delay timer. *(Courtesy of Allen-Bradley Company, Inc.)*

Instruction list

Temporized Relays

or

Timers

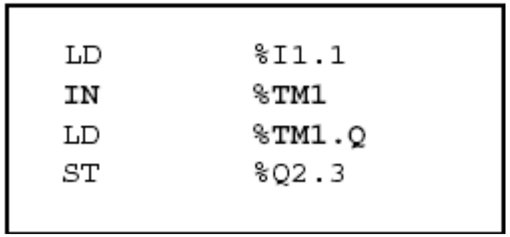


Characteristics:

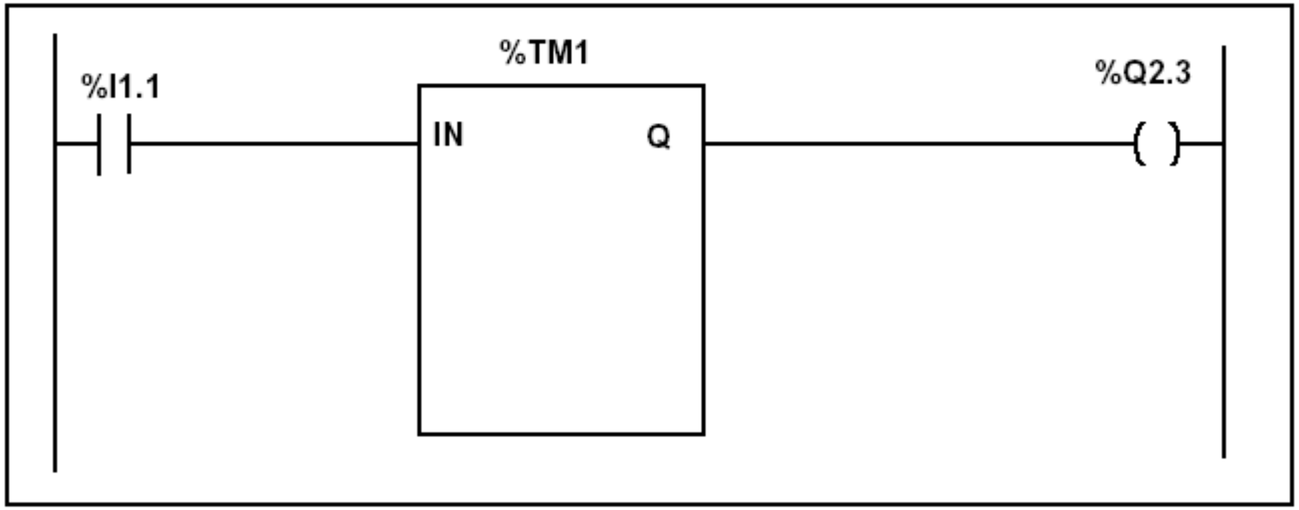
Identifier:%TMi	0..63 in the TSX37	
Input:	IN	to activate
Mode:	TON TOFF TP	On delay Off delay Monostable
Time basis:	TB	1mn (def.), 1s, 100ms, 10ms
Programmed value:	%TMi.P	0...9999 (def.) period=TB*TMi.P
Actual value:	%TMi.V	0...TMi.P (can be real or tested)
Modifiable:	Y/N	can be modified from the console

Instruction list

Relés temporizados
Ou
Timers

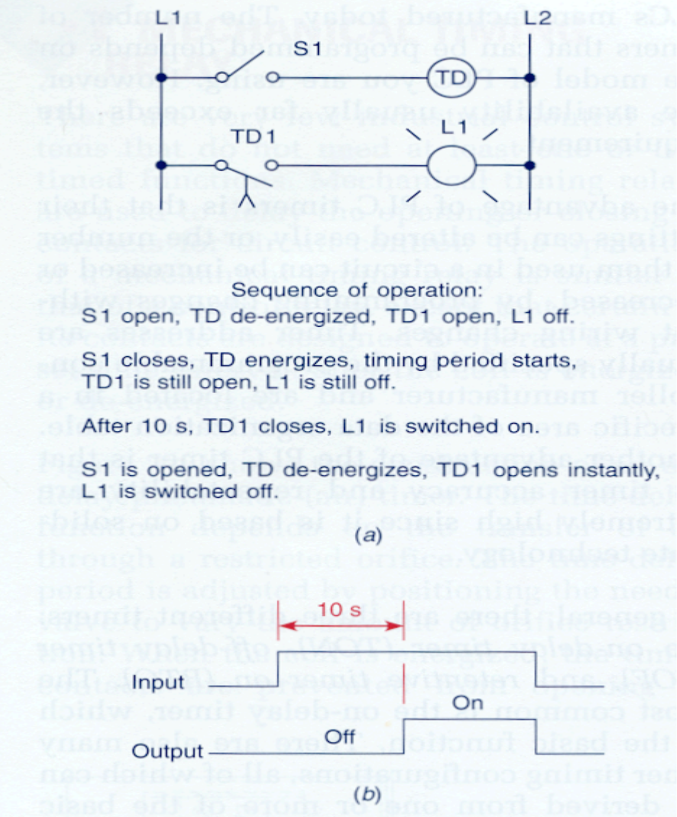


```
LD          %I1.1
IN          %TM1
LD          %TM1.Q
ST          %Q2.3
```



Instruction list

Example:



Sequence of operation:

S1 open, TD de-energized, TD1 open, L1 off.

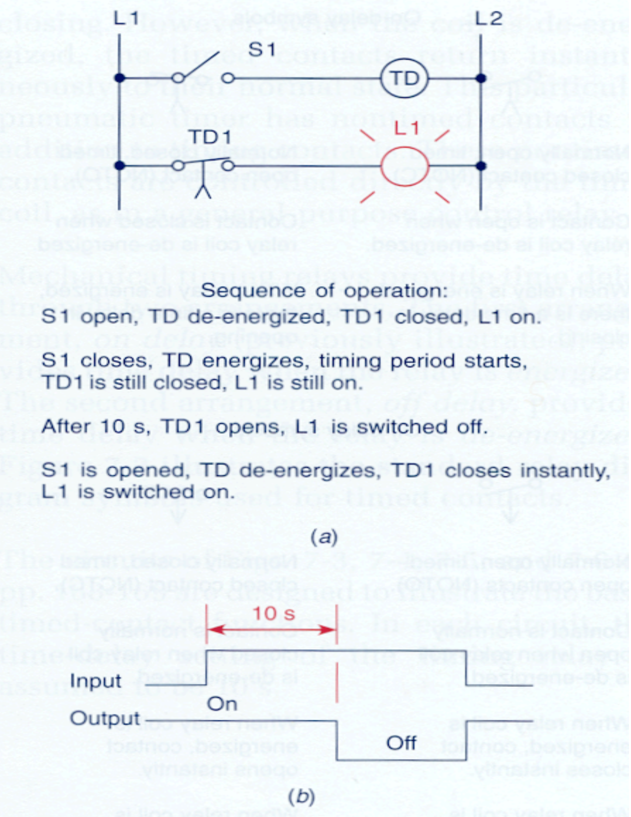
S1 closes, TD energizes, timing period starts, TD1 is still open, L1 is still off.

After 10 s, TD1 closes, L1 is switched on.

S1 is opened, TD de-energizes, TD1 opens instantly, L1 is switched off.

Fig. 7-3

On-delay timer circuit (NOTC contact). (a) Operation. (b) Timing diagram.



Sequence of operation:

S1 open, TD de-energized, TD1 closed, L1 on.

S1 closes, TD energizes, timing period starts, TD1 is still closed, L1 is still on.

After 10 s, TD1 opens, L1 is switched off.

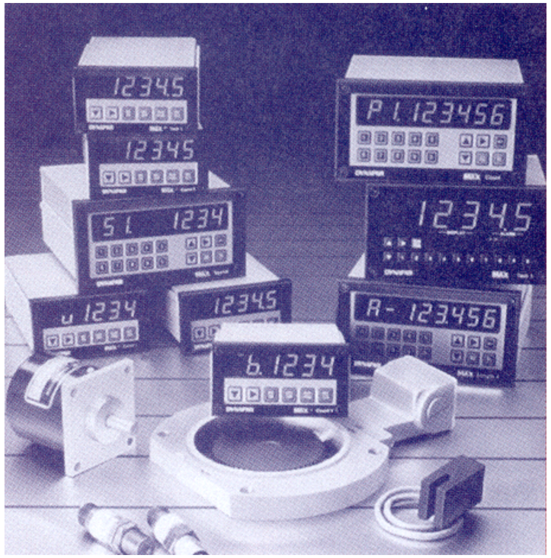
S1 is opened, TD de-energizes, TD1 closes instantly, L1 is switched on.

Fig. 7-4

On-delay timer circuit (NCTO contact). (a) Operation. (b) Timing diagram.

Instruction list

Counters



Some applications...

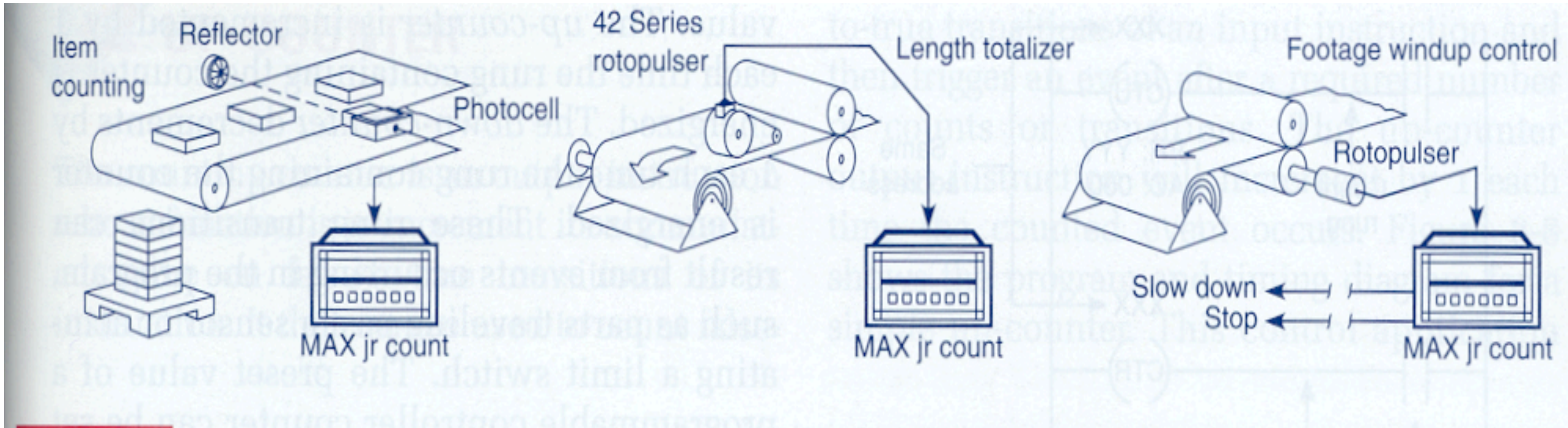
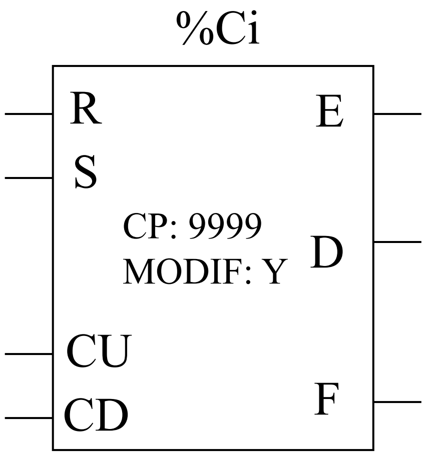


Fig. 8-3

Counter applications. *(Courtesy of Dynapar Corporation, Gurnee, Illinois.)*

Instruction list

Counters



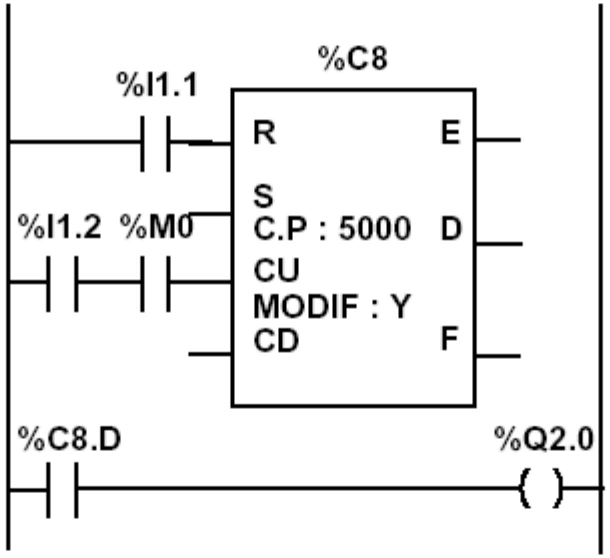
Characteristics:

Identifier:%Ci	0..31 in the TSX37	
Value progr.:	%Ci.P	0...9999 (def.)
Value Actual:	%Ci.V	0...Ci.P (only to be read)
Modifiable:	Y/N	can be modified from the console
Inputs:	R	Reset Ci.V=0
	S	Preset Ci.V=Ci.P
	CU	*Count Up*
	CD	*Count Down*
Outputs:	E	Overrun %Ci.E=1 %Ci.V=0->9999
	D	Done %Ci.D=1 %Ci.V=Ci.P
	F	Full %Ci.F=1 %Ci.V=9999->0

Instruction list

Counters

Example:



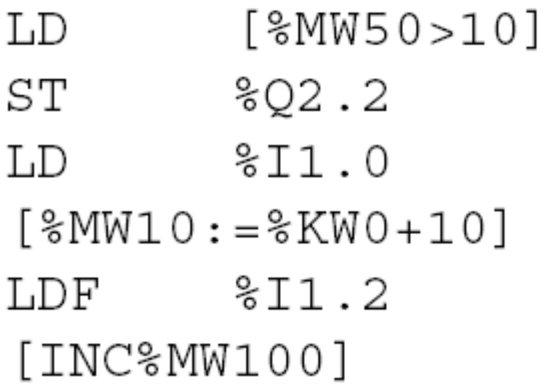
Instruction list language

```
LD %I1.1
R   %C8
LD  %I1.2
AND %M0
CU  %C8
LD  %C8.D
ST  %Q2.0
```

Instruction list

Numerical Processing

Algebraic and Logic Functions

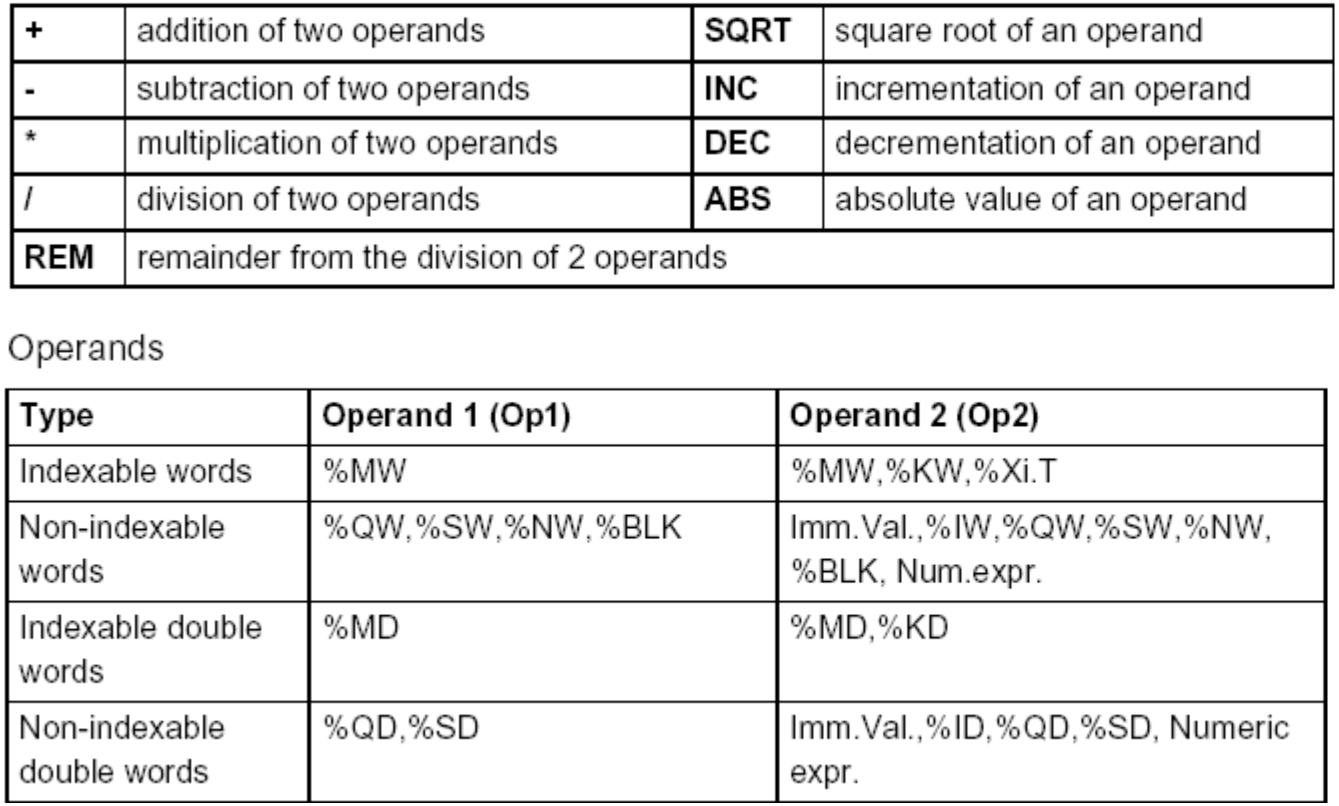


```
LD      [%MW50>10]
ST      %Q2.2
LD      %I1.0
[%MW10:=%KW0+10]
LDF     %I1.2
[INC%MW100]
```

Instruction list

Numerical Processing

Arithmetic Functions



+	addition of two operands	**SQRT**	square root of an operand
-	subtraction of two operands	**INC**	incrementation of an operand
*	multiplication of two operands	**DEC**	decrementation of an operand
/	division of two operands	**ABS**	absolute value of an operand
REM	remainder from the division of 2 operands		

Operands

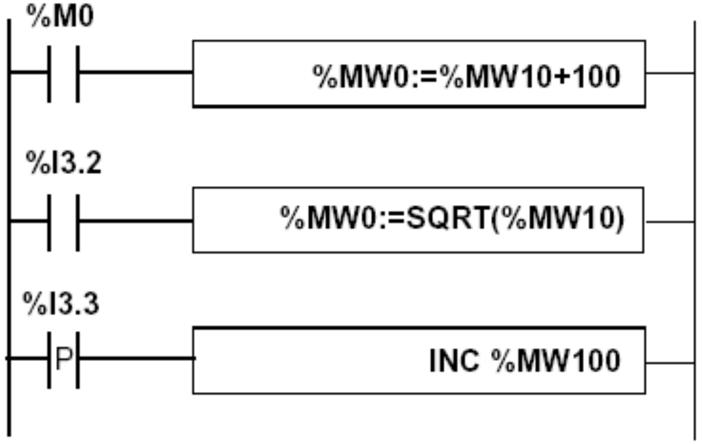
Type	**Operand 1 (Op1)**	**Operand 2 (Op2)**
Indexable words	%MW	%MW,%KW,%Xi.T
Non-indexable words	%QW,%SW,%NW,%BLK	Imm.Val.,%IW,%QW,%SW,%NW, %BLK, Num.expr.
Indexable double words	%MD	%MD,%KD
Non-indexable double words	%QD,%SD	Imm.Val.,%ID,%QD,%SD, Numeric expr.

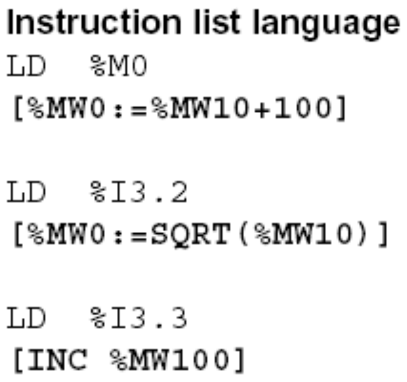
Instruction list

Numerical Processing

Example:

Arithmetic functions





Instruction list language

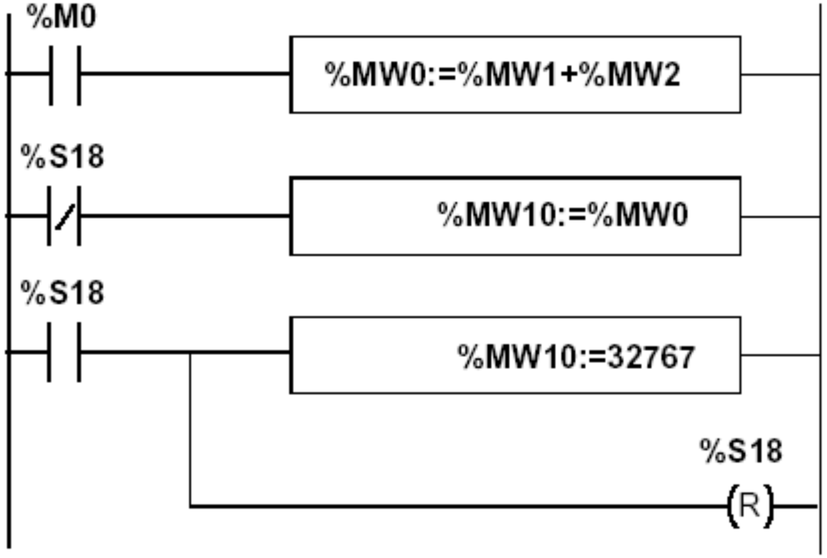
```
LD  %M0
[%MW0:=%MW10+100]

LD  %I3.2
[%MW0:=SQRT(%MW10)]

LD  %I3.3
[INC %MW100]
```

Instruction list

Numerical Processing

Example:

Arithmetic functions



Example in instruction list language:

```
LD    %M0
[%MW0:=%MW1+%MW2]
LDN   %S18
[%MW10:=%MW0]
LD    %S18
[%MW10:=32767]
R    %S18]
```

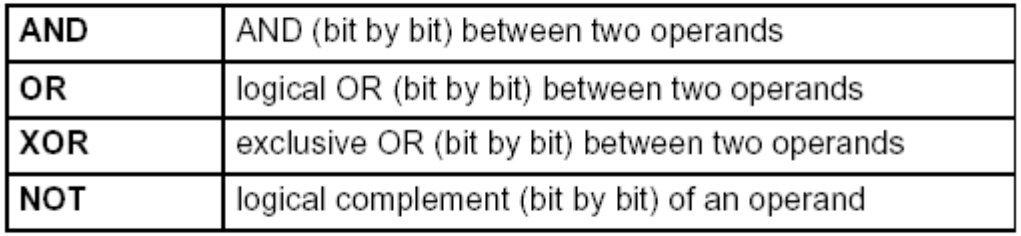
Use of a system variable:

%S18 – flag de overflow

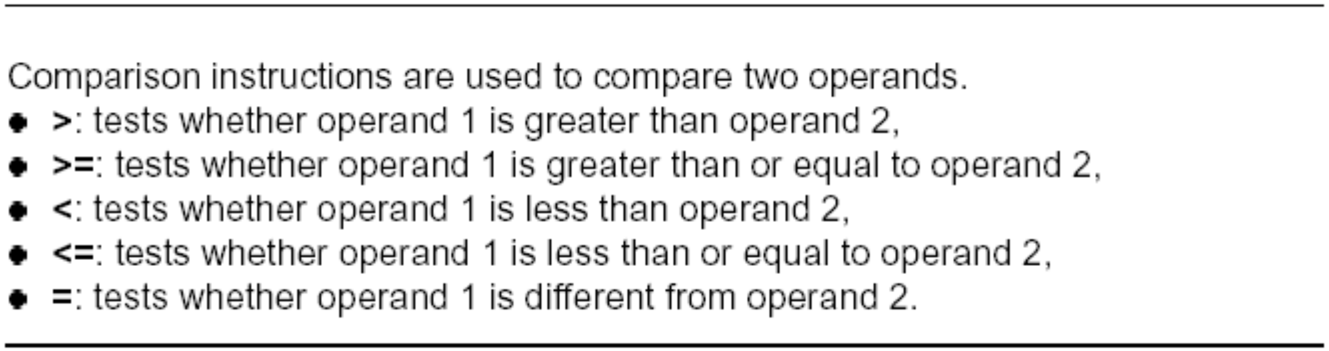
Instruction list

Numerical Processing

Logic Functions

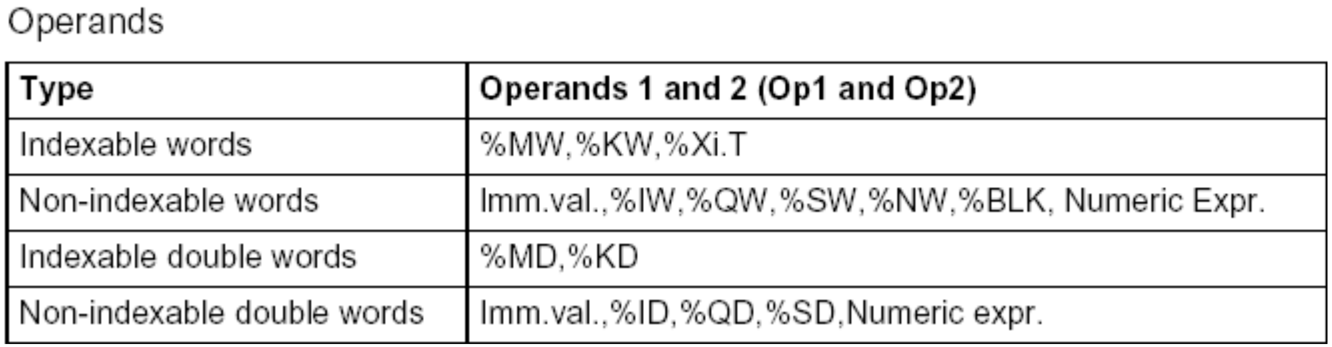


AND	AND (bit by bit) between two operands
OR	logical OR (bit by bit) between two operands
XOR	exclusive OR (bit by bit) between two operands
NOT	logical complement (bit by bit) of an operand



Comparison instructions are used to compare two operands.

- **>**: tests whether operand 1 is greater than operand 2,
- **>=**: tests whether operand 1 is greater than or equal to operand 2,
- **<**: tests whether operand 1 is less than operand 2,
- **<=**: tests whether operand 1 is less than or equal to operand 2,
- **=**: tests whether operand 1 is different from operand 2.



Operands

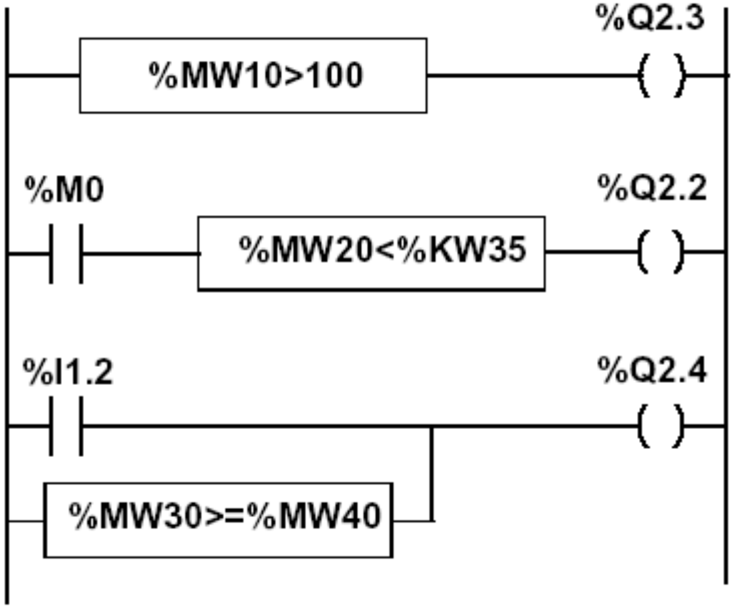
Type	**Operands 1 and 2 (Op1 and Op2)**
Indexable words	%MW,%KW,%Xi.T
Non-indexable words	Imm.val.,%IW,%QW,%SW,%NW,%BLK, Numeric Expr.
Indexable double words	%MD,%KD
Non-indexable double words	Imm.val.,%ID,%QD,%SD,Numeric expr.

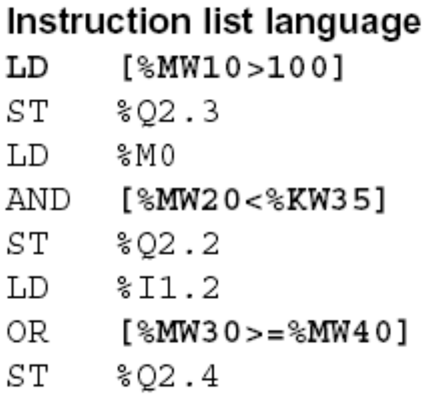
Instruction list

Numerical Processing

Example:

Logic functions





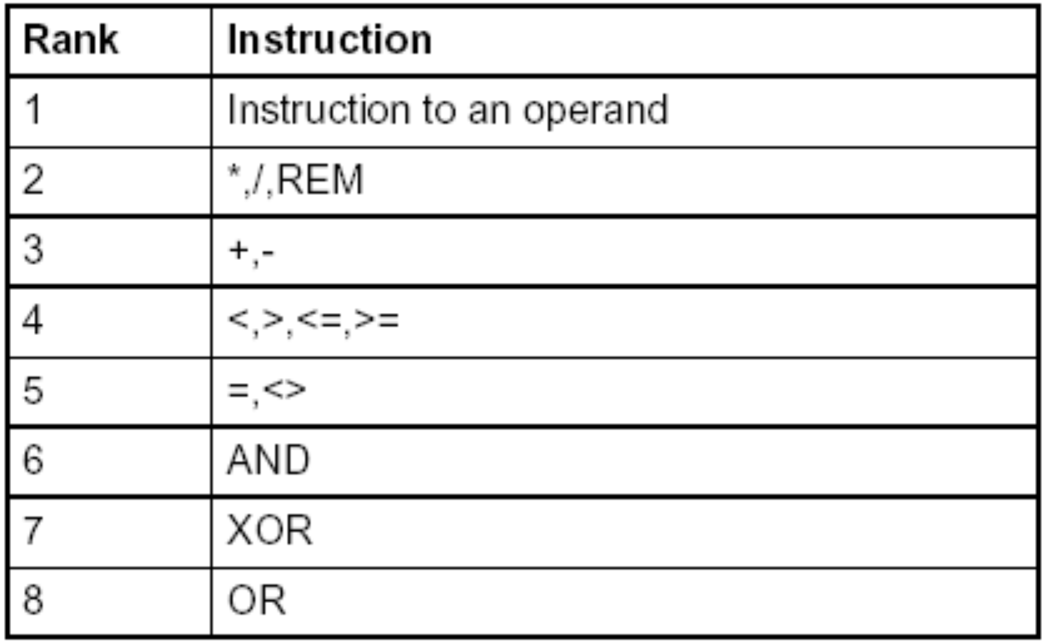
Instruction list language

```
LD    [%MW10>100]
ST    %Q2.3
LD    %M0
AND   [%MW20<%KW35]
ST    %Q2.2
LD    %I1.2
OR    [%MW30>=%MW40]
ST    %Q2.4
```

Instruction list

Numerical Processing

Priorities on the execution of the operations



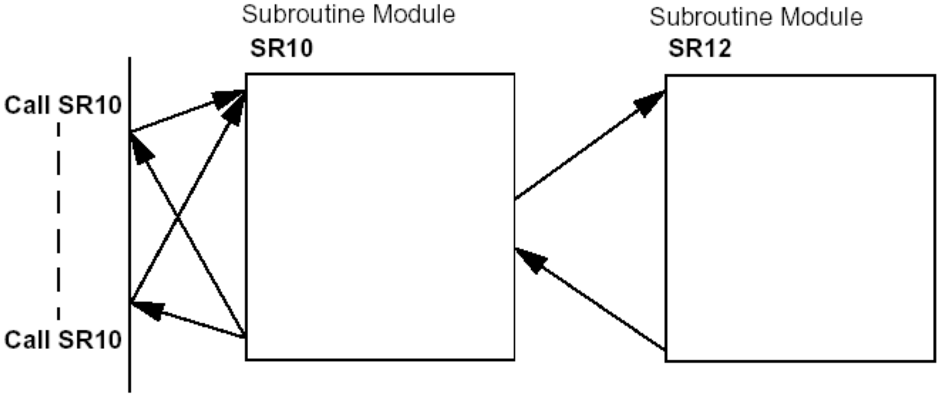
Rank	Instruction
1	Instruction to an operand
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

Instruction list

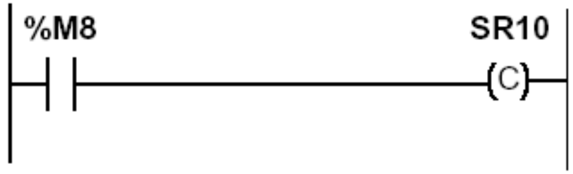
Structures for Control of Flux

Subroutines

Call and Return



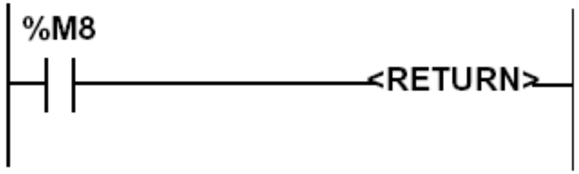
Ladder language:



Instruction list language:

```
LD   %M8
SR10
```

Ladder language



Instruction list language

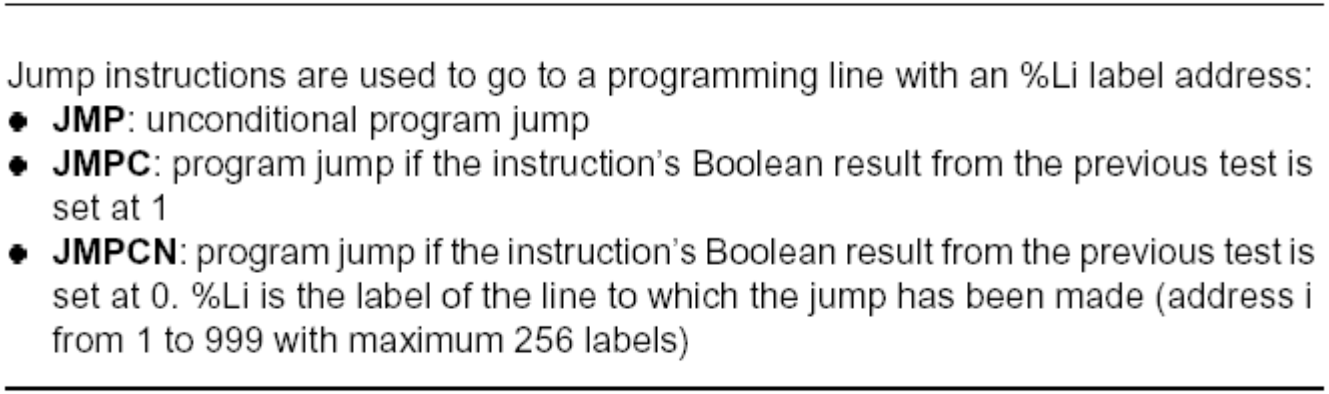
```
LD   %M8
RETC
```

Instruction list

Structures for Control of Flux

JUMP instructions:

Conditional and unconditional



Jump instructions are used to go to a programming line with an %Li label address:

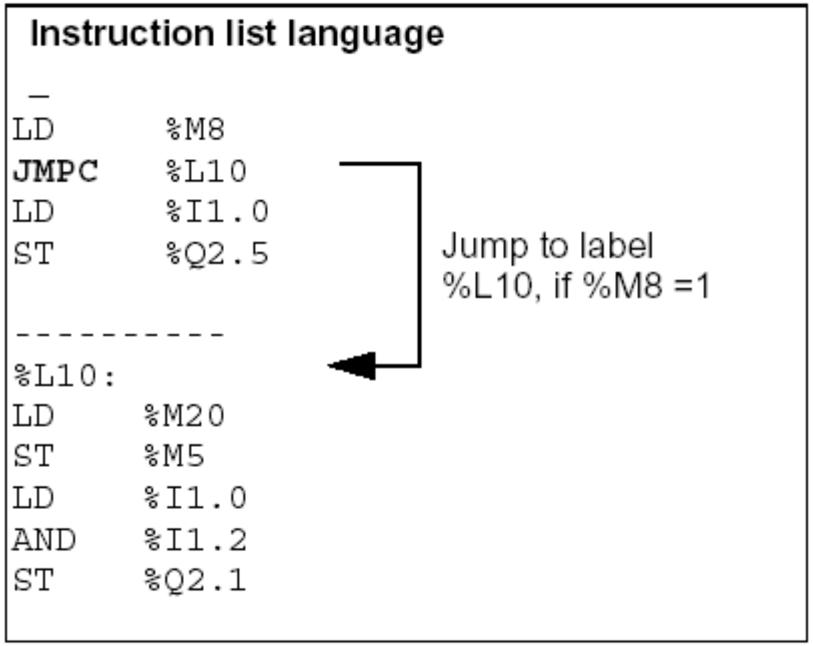
- **JMP**: unconditional program jump
- **JMPC**: program jump if the instruction's Boolean result from the previous test is set at 1
- **JMPCN**: program jump if the instruction's Boolean result from the previous test is set at 0. %Li is the label of the line to which the jump has been made (address i from 1 to 999 with maximum 256 labels)

Instruction list

Structures for Control of Flux

Example:

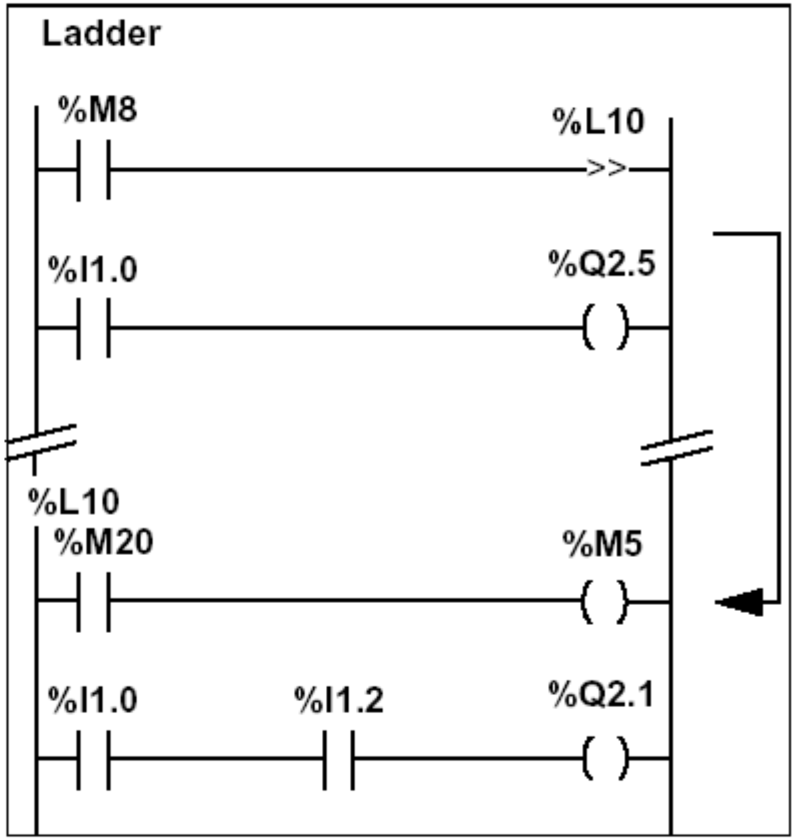
Use of jump instructions



Instruction list language

```
 _
LD      %M8
JMPC    %L10
LD      %I1.0
ST      %Q2.5

----------
%L10:
LD     %M20
ST     %M5
LD     %I1.0
AND    %I1.2
ST     %Q2.1
```

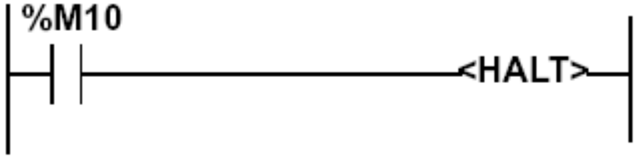
Jump to label
%L10, if %M8 =1



Instruction list

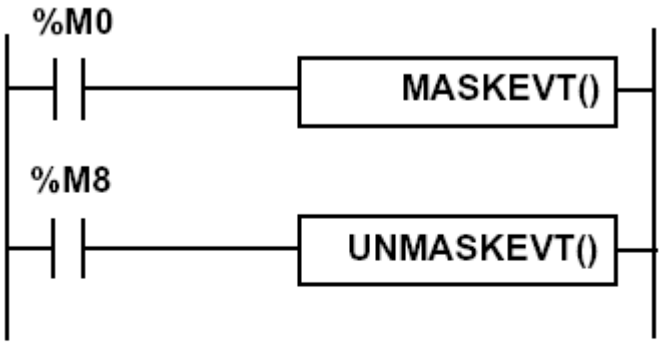
Structures for Control of Flux

Halt



Stops all processes!

Events masking



Instruction list

There are other advanced instrauctions (see manual)

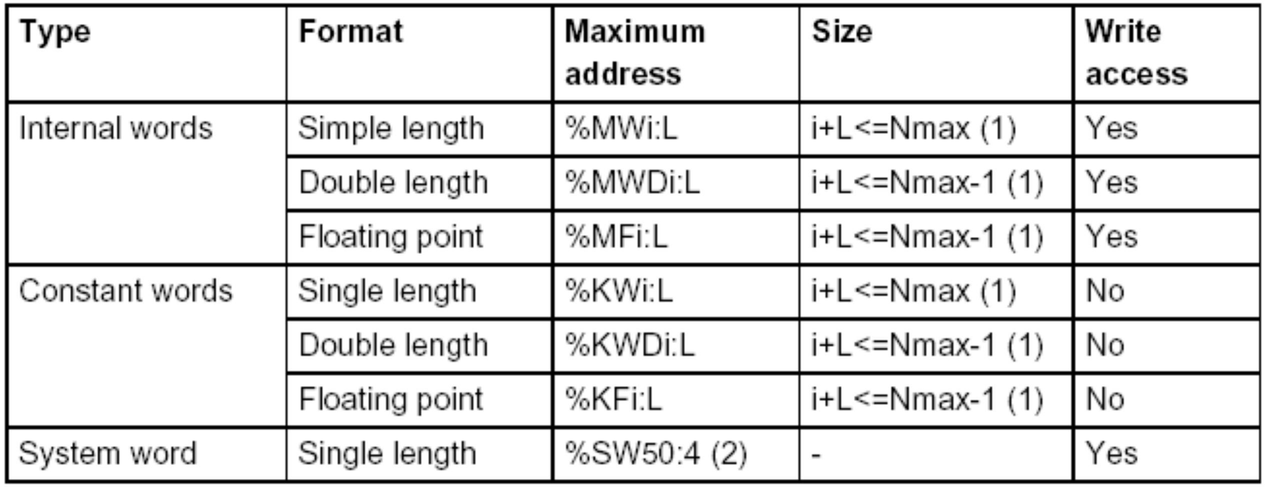
- **Monostable**
- **Registers of 256 words (LIFO ou FIFO)**
- ***DRUMs***
- **Comparators**
- ***Shift-registers***

...

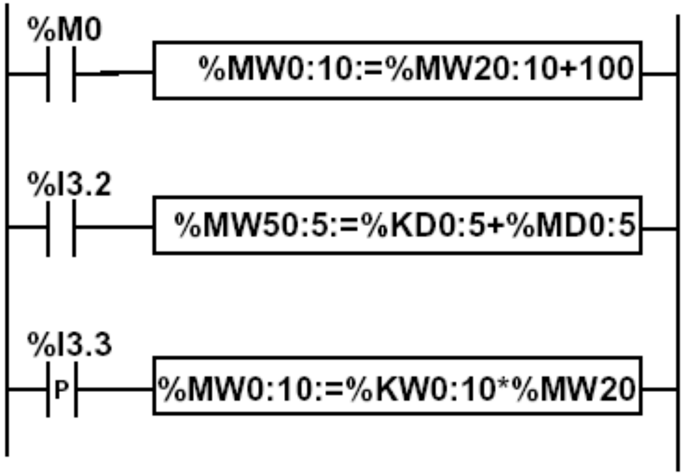
- **Functions to manipulate *floats***
- **Functions to convert bases and types**

Instruction list

Numerical Tables



Type	Format	Maximum address	Size	Write access
Internal words	Simple length	%MWi:L	i+L<=Nmax (1)	Yes
	Double length	%MWDi:L	i+L<=Nmax-1 (1)	Yes
	Floating point	%MFi:L	i+L<=Nmax-1 (1)	Yes
Constant words	Single length	%KWi:L	i+L<=Nmax (1)	No
	Double length	%KWDi:L	i+L<=Nmax-1 (1)	No
	Floating point	%KFi:L	i+L<=Nmax-1 (1)	No
System word	Single length	%SW50:4 (2)	-	Yes



Instruction list language

```
LD %M0
[%MW0:10:=%MW20:10+100]

LD %I3.2
[%MD50:5:=%KD0:5+%MD0:5]
```

DOLOG80

PLC AEG A020 Plus:

Inputs:
- 20 binary with opto-couplers
- 4 analogs (8 bits, 0-10V)

Outputs:
- 16 binary with relays of 2A
- 1 analogs (8 bits, 0-10V)

Interface for progr.: RS232

Processador:
- 8031
- 2 Kbytes de RAM
- 2 Kbytes EEPROM => 896 instructions
- **Average cycle time: 6.5 ms**

PLC AEG A020 Plus

DOLOG80

OPERANDS

• I1 to I20	Binary inputs
• Q1 to Q16	Bynary outputs
• M1 to M128	Auxiliary memory
• T1 to T8	*Timers* (base 100ms)
• T9 to T16	*Timers* (base 25ms)
• C1 to C16	Contadores with16 *bits*

DOLOG80 (cont.)

Example:

```
AI1
A(
OI2
O(
ANC9
AQ9
)
)
=Q9
...
AI3
=P9
NO
OM1
OI4
=Z9
NO
AC9
=M1
...
LDV50
=CSW9
PE
```

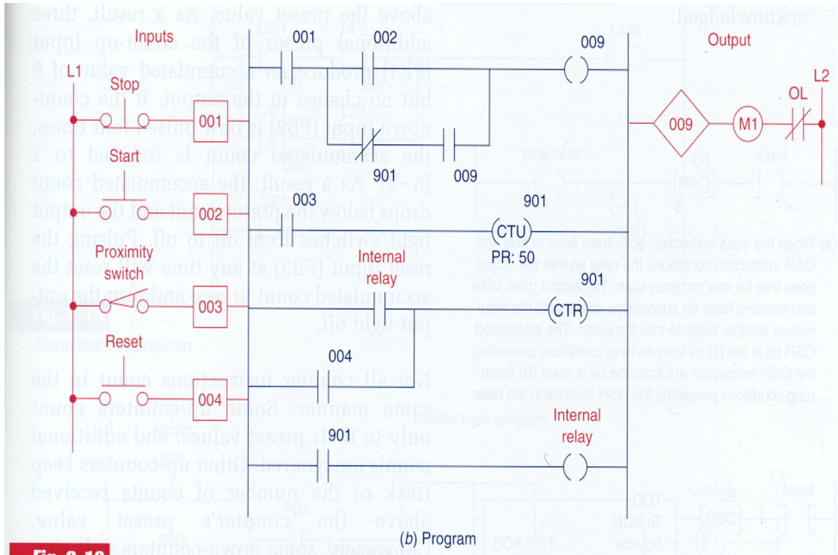


Fig. 8-13 Conveyor motor program.

Legend:

Stop = I1
Start = I2
Proximity Sensor = I3
Reset = I4
Counter= C9
Internal relay = M1
Motor = Q9