Industrial Automation

(Automação de Processos Industriais)

PLCs Programming Languages

Structured Text

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*.

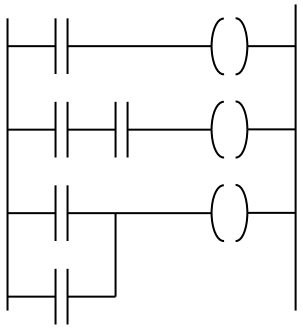
Software development resources.

...

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

PLCs Programming Languages (IEC 1131-3)

Ladder Diagram



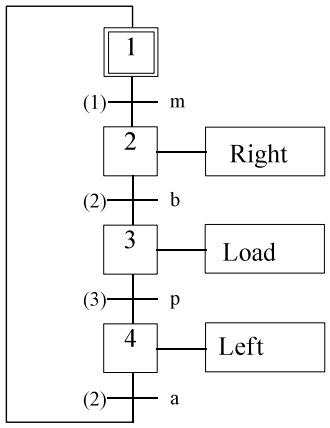
Structured Text

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

Instruction List

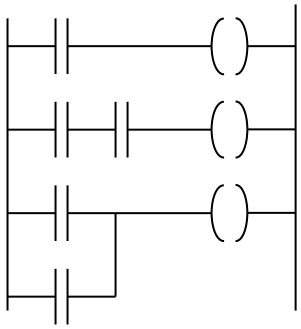
```
LD      %M12
AND     %I1.0
ANDN    %I1.1
OR      %M10
ST      %Q2.0
```

Sequential Function Chart (GRAFCET)



Linguagens de programação de PLCs (IEC 1131-3)

Ladder Diagram



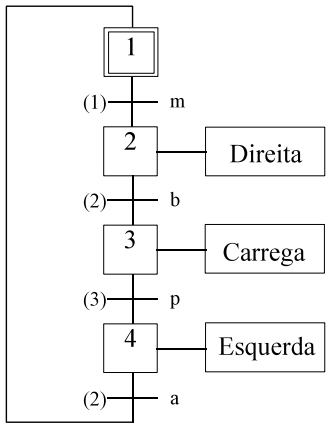
Structured Text

```
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)



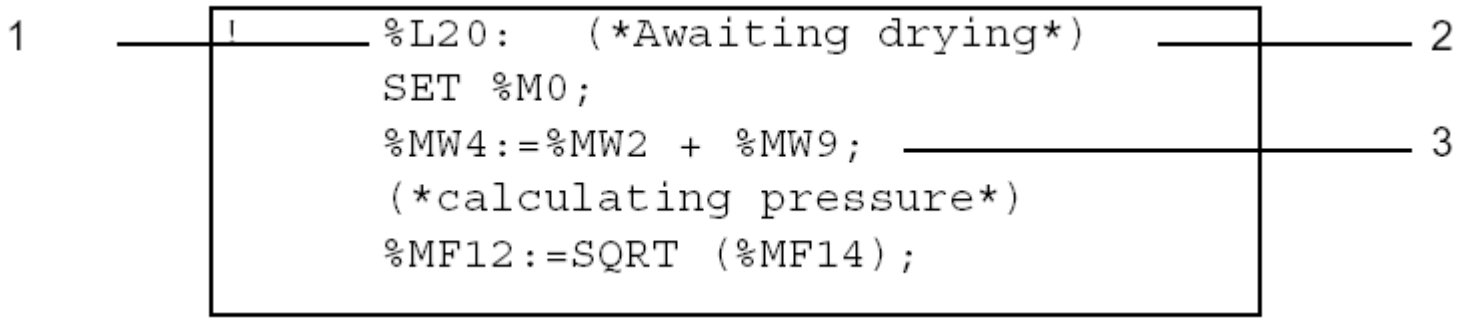
Structured Text

```
!     (* Searching for the first element that is not zero
in a
      table of 32 words, determining its value
      (%MW10), its rank (%MW11). This search
      is done if %M0 is set to 1, %M1 is set to 1 if
      an element which is not zero exists unless it is
set to 0*)

      IF %M0 THEN
            FOR %MW99:=0 TO 31 DO
                  IF %MW100[%MW99]<>0 THEN
                        %MW10:=%MW100[%MW99];
                        %MW11:=%MW99;
                        %M1:=TRUE;
                        EXIT;    (*Exit the loop*)
                  ELSE
                        %M1:=FALSE;
                  END_IF;
            END_FOR;
      ELSE
            %M1:=FALSE;
      END_IF;
```

Structured Text

A section of the program is composed by sequences

One sequence is equivalent to a section in *ladder diagram* (one or more rangs).



Legend:

1 – label - unique identifier (%Li, i=0...999)

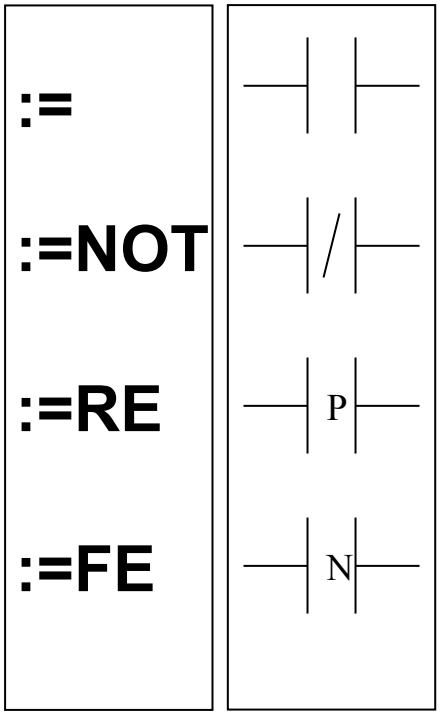
2 – comments - augments legibility (* limited to 256 bytes *)

3 – instructions

Structured Text

Basic Instructions

Load



Open contact: contact is active (result is 1) while the control bit is 1.

Close contact: contacto is active (result is 1) while the control bit is 0.

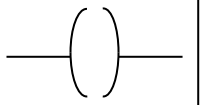
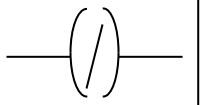
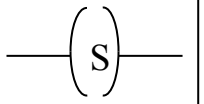
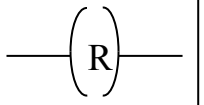
Contact in the rising edge: contact is active during a scan cycle where the control bit has a rising edge.

Contact in the falling edge: contact is active during a scan cycle where the control bit has a falling edge.

Structured Text

Basic Instructions

Store

:=		The result of the logic function activates the coil.
:=NOT		The inverse result of the logic function activates the coil.
SET		The result of the logic function energizes the relay (sets the latch).
RESET		The result of the logic function de-energizes the relay (resets the latch)..

Structured Text

Basic Instructions

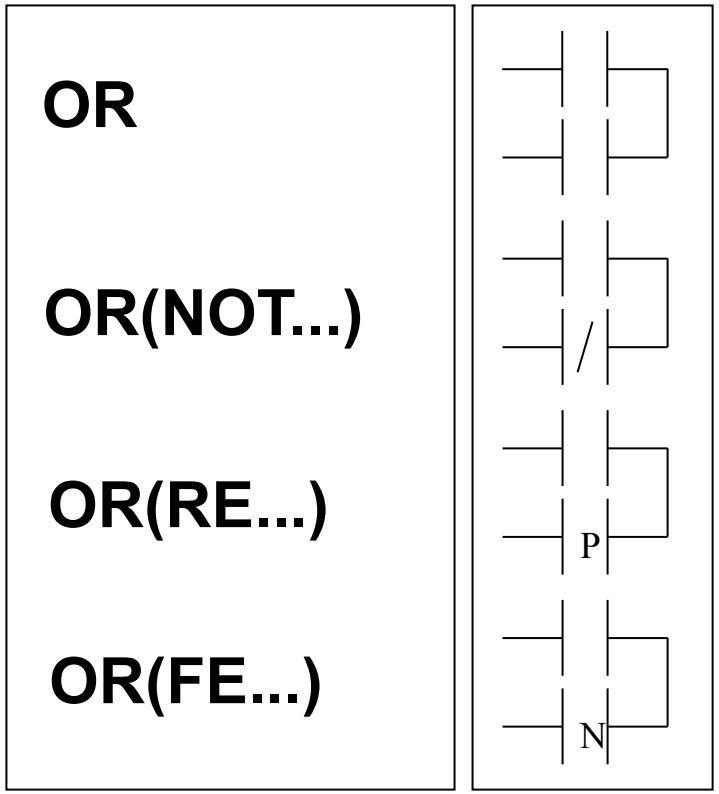
AND

AND	AND of the operand with the result of the previous logical operation.
AND(NOT...)	AND of the operand with the inverted result of the previous logical operation.
AND(RE...)	AND of the rising edge with the result of the previous logical operation.
AND(FE...)	AND of the falling edge with the result of the previous logical operation.

Structured Text

Basic Instructions

OR



OR of the operand with the result of the previous logical operation.

OR of the operand with the inverted result of the previous logical operation.

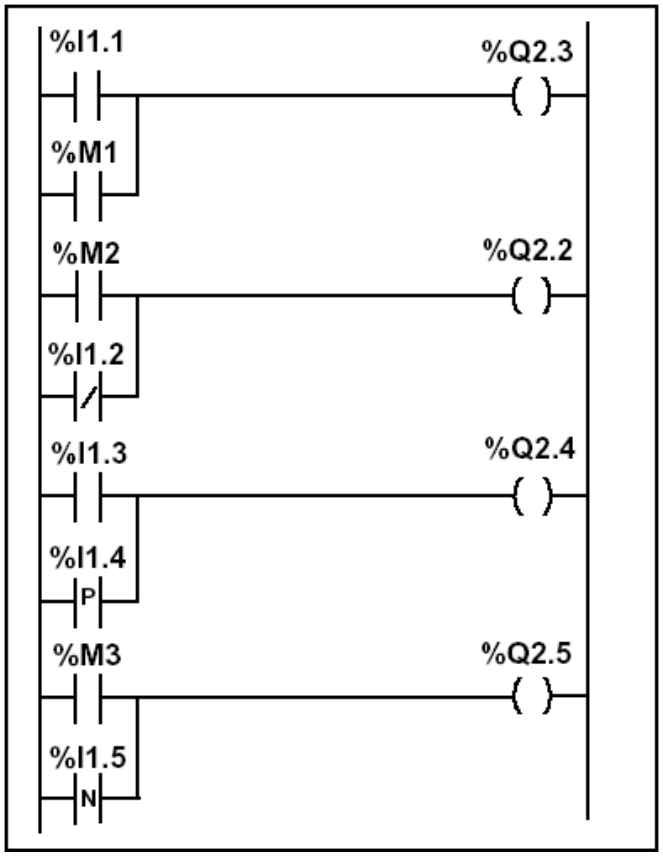
OR of the rising edge with the result of the previous logical operation.

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

Structured Text

Example:

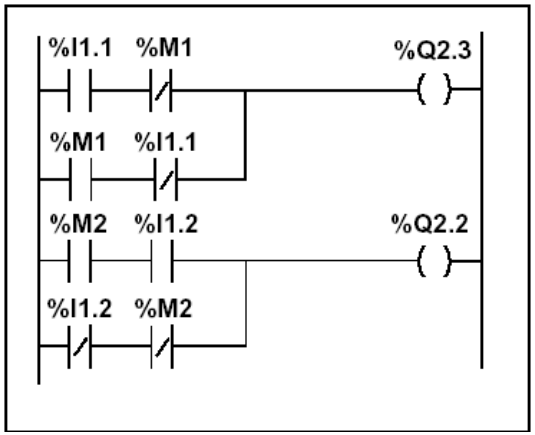
```
%Q2.3:=%I1.1 OR %M1;
%Q2.2:=%M2 OR (NOT%I1.2);
%Q2.4:=%I1.3 OR (RE%I1.4);
%Q2.5:=%M3 OR (FE%I1.5);
```



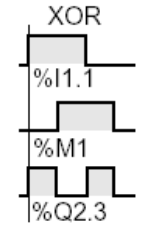
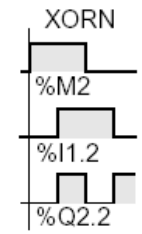
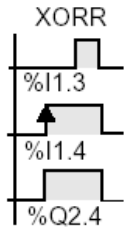
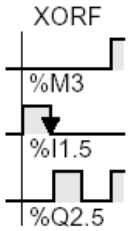
Structured Text

Basic Instructions

XOR




```
%Q2.3:=%I1.1 XOR%M1;
%Q2.2:=%M2 XOR (NOT%I1.2);
%Q2.4:=%I1.3 XOR (RE%I1.4)
%Q2.5:=%M3 XOR (FE%I1.5);
```

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.	

Structured Text

Basic Instructions to Manipulate Bit Tables

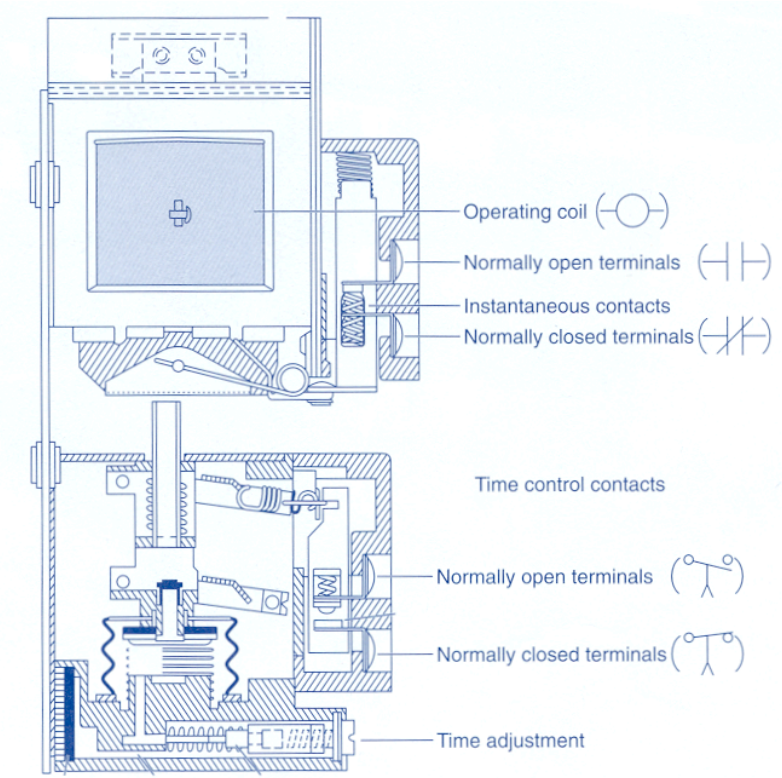
Designation	Function
Table:= Table	Assignment between two tables
Table:= Word	Assignment of a word to a table
Word:= Table	Assignment of a table to a word
Table:= Double word	Assignment of a double word to a table
Double word: = Table	Assignment of a table to a double word
COPY_BIT	Copy of a bits table in a bits table
AND_ARX	AND between two tables
OR_ARX	OR between two tables
XOR_ARX	exclusive OR between two tables
NOT_ARX	Negation in a table
BIT_W	Copy of a bits table in a word table
BIT_D	Copy of a bits table in a double word table
W_BIT	Copy of a word table in a bits table
D_BIT	Copy of a double word table in a bits table
LENGHT_ARX	Calculation of the length of a table by the number of elements

Structured Text

Temporized Relays

or

Timers



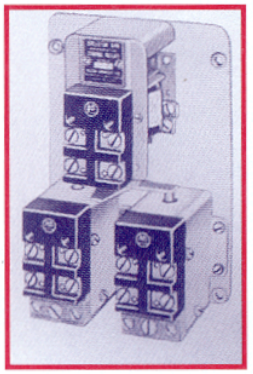


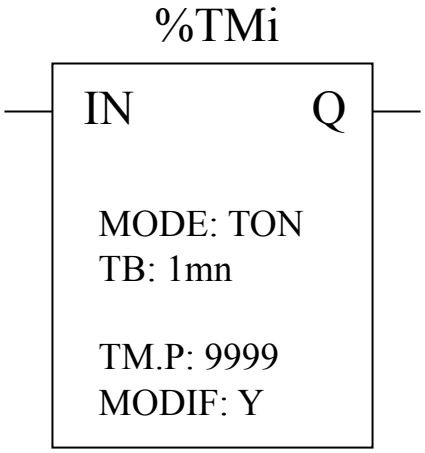
Fig. 7-1
Pneumatic on-delay timer. *(Courtesy of Allen-Bradley Company, Inc.)*

Structured Text

Temporized Relays

or

Timers

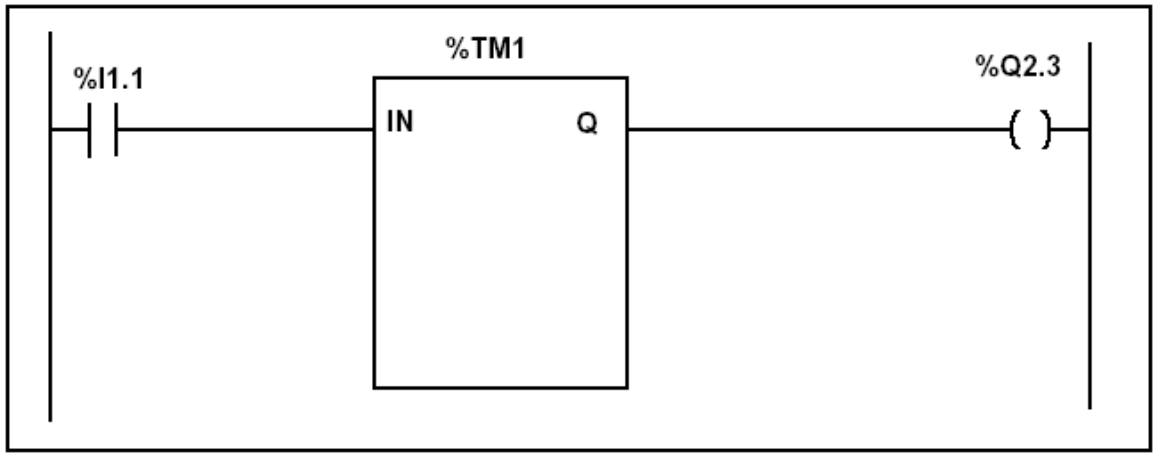


Characteristics:

Identifier:%TMi	0..63 in the TSX37	
Input:	IN	to activate
Mode:	TON	On delay
	TOFF	Off delay
	TP	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

Structured Text

Relés temporizados
Ou
Timers




```
IF %I1.1 THEN
        START %TM1;
END_IF;
%Q2.3 := %TM1.Q
```

Structured Text

Example:

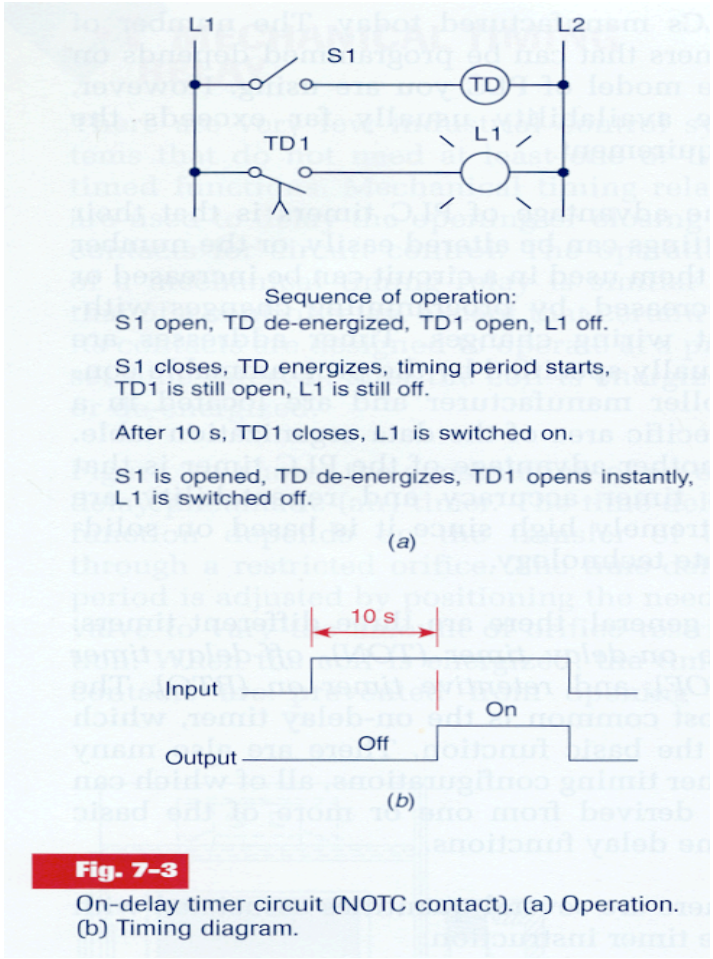


Fig. 7-3
On-delay timer circuit (NOTC contact). (a) Operation. (b) Timing diagram.

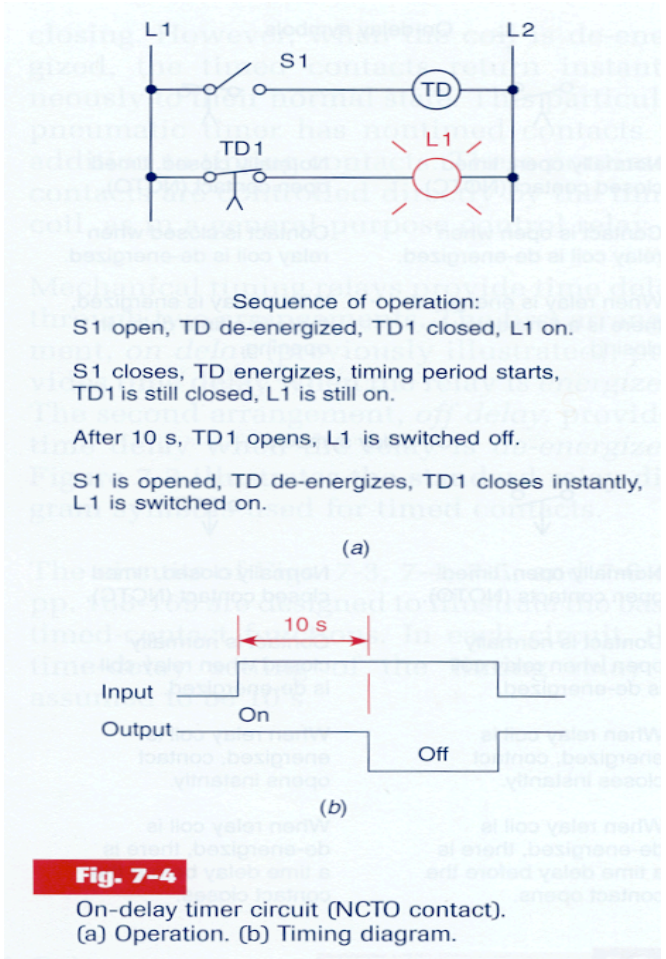
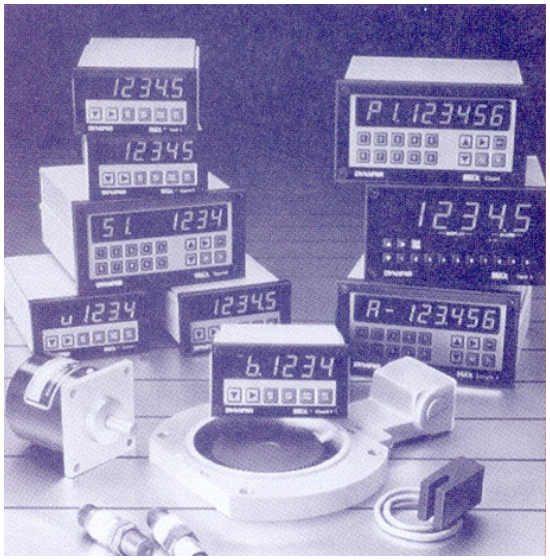


Fig. 7-4
On-delay timer circuit (NCTO contact). (a) Operation. (b) Timing diagram.

Structured Text

Counters



Some applications...

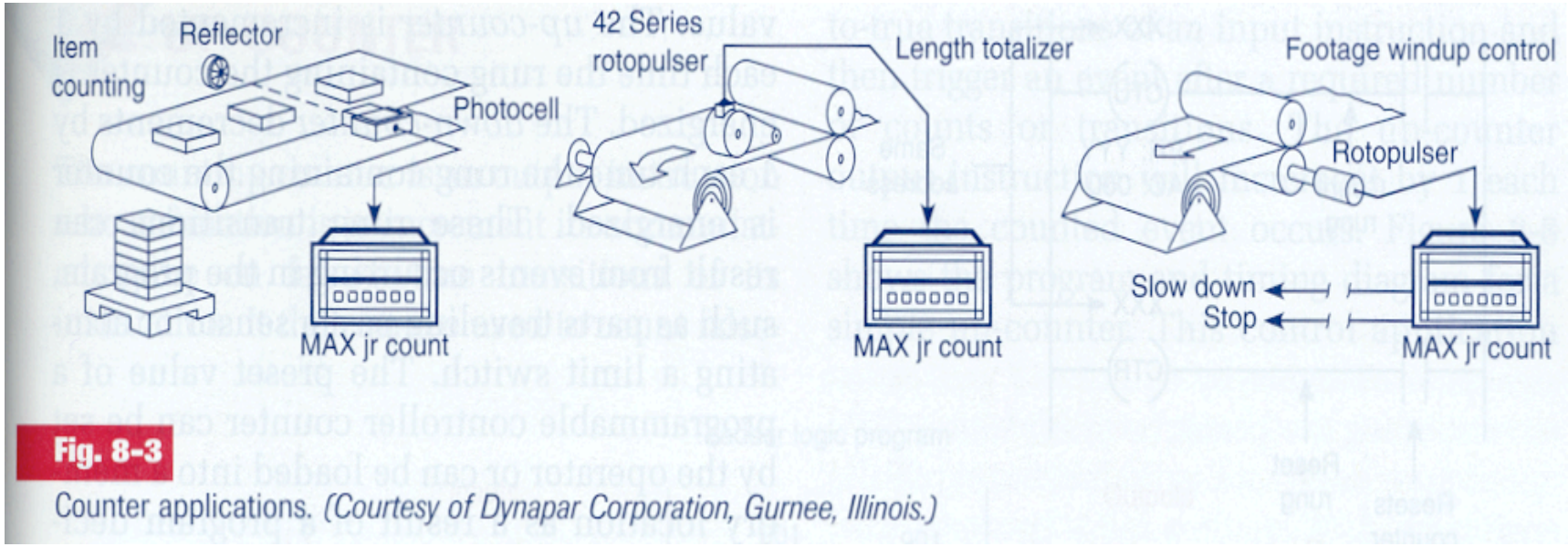
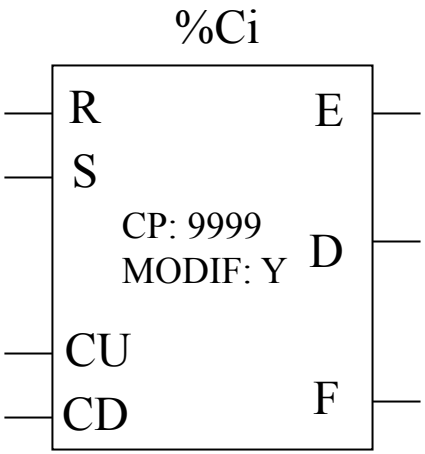


Fig. 8-3
Counter applications. *(Courtesy of Dynapar Corporation, Gurnee, Illinois.)*

Structured Text

Counters



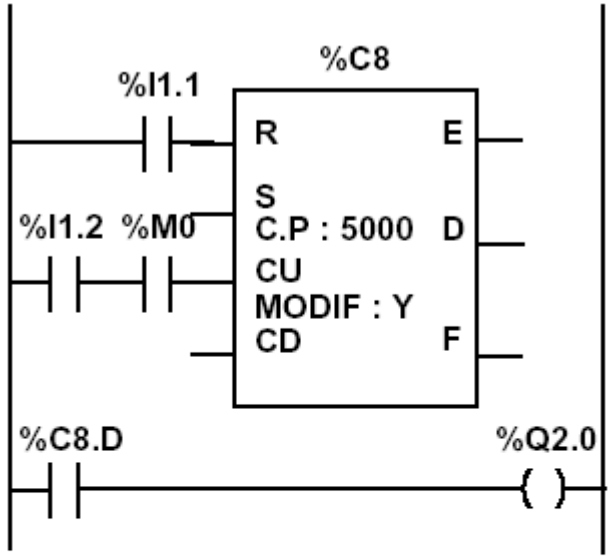
Characteristics:

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

Structured Text

Counters

Example:




```
IF %I1.1 THEN
        RESET %C8;
END_IF;
IF (%I1.2 AND %M0) THEN
        UP %C8;
END_IF;
%Q2.0 := %C8.D;
```

Instruction list language

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

Structured Text

Numerical Processing

Algebraic and Logic Functions

```
%Q2.2:=%MW50 > 10;
IF %I1.0 THEN
   %MW10:=%KW0 + 10;
END_IF;
IF FE %I1.2 THEN
   INC %MW100;
END_IF;
```

Structured Text

Numerical Processing

Arithmetic Functions for Words

+	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.

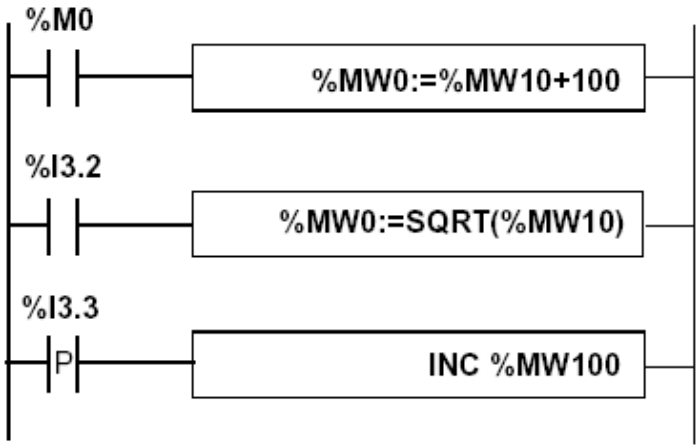
Structured Text

Numerical Processing

Example:

Arithmetic functions

```
IF %M0 THEN
        %MW0 := %MW10 + 100;
END_IF;
IF %I3.2 THEN
         %MW0 := SQRT(%MW10);
END_IF;
IF RE %I3.3 THEN
        INC %MW100;
END_IF;
```



Instruction list language

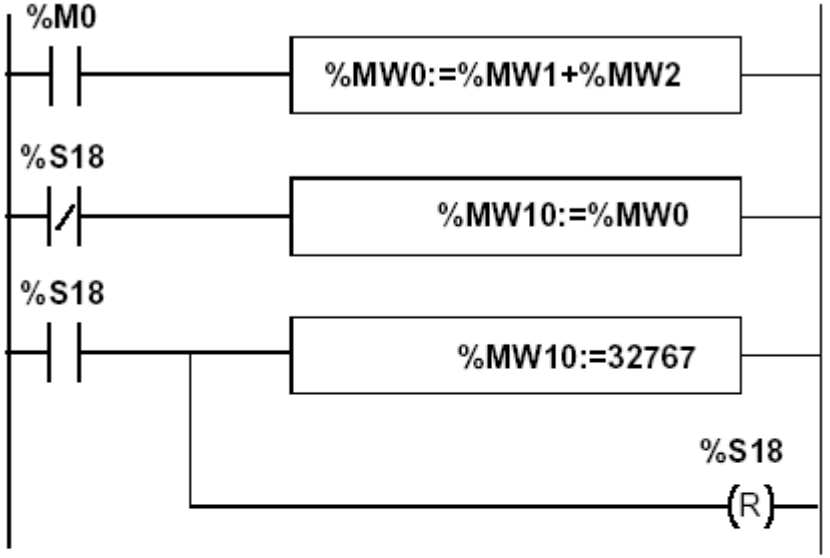
```
LD   %M0
[%MW0:=%MW10+100]

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

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

Structured Text

Numerical Processing

Example:

Arithmetic functions




```
IF %M0 THEN
        %MW0 := %MW1 + %MW2;
END_IF;
IF %S18 THEN
        %MW10 := 32767; RESET %S18;
ELSE
        %MW10 := %MW0;
END_IF;
```

Example in instruction list language:

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

Structured Text

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

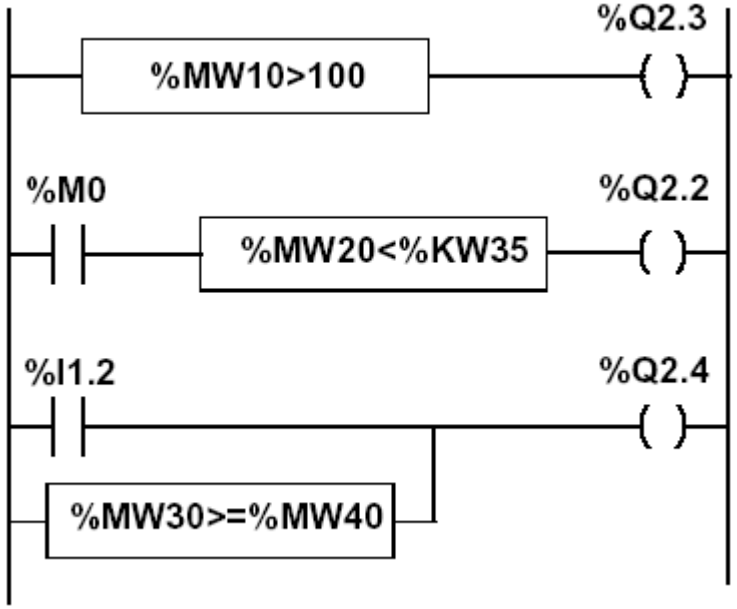
Type	**Operands 1 and 2 (Op1 and Op2)**
Indexable words	%MW,%KW,%Xi.T
Non-indexable words	Imm.val.,%IW,%QW,%SW,%NW,%BLK, Numeric Expr.
Indexable double words	%MD,%KD
Non-indexable double words	Imm.val.,%ID,%QD,%SD,Numeric expr.

Structured Text

Numerical Processing

Example:

Logic functions



Structured text language

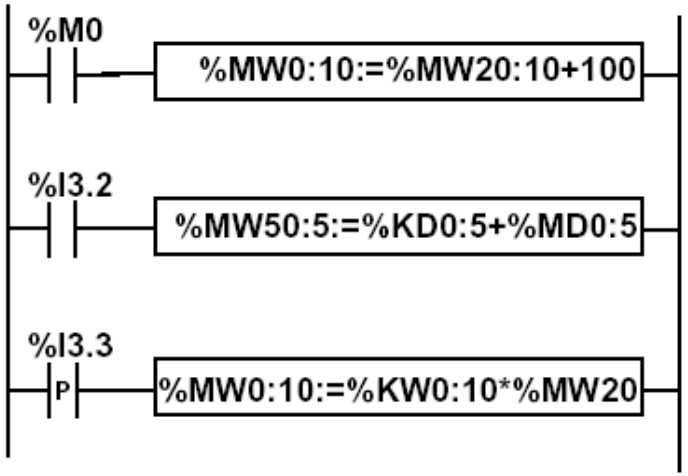
```
%Q2.3:=%MW10>100;
%Q2.2:=%M0 AND (%MW20<%KW35);
%Q2.4:=%I1.2 OR (%MW30>=%MW40);
```

Structured Text

Numerical Processing

Example:

Numeric Tables Manipulation



Structured text language

```
IF RE %I3.3 THEN
 %MW0:10:=%KW0:10*%MW20;
END_IF;
```

Structured Text

Numerical Processing

Priorities on the execution of the operations

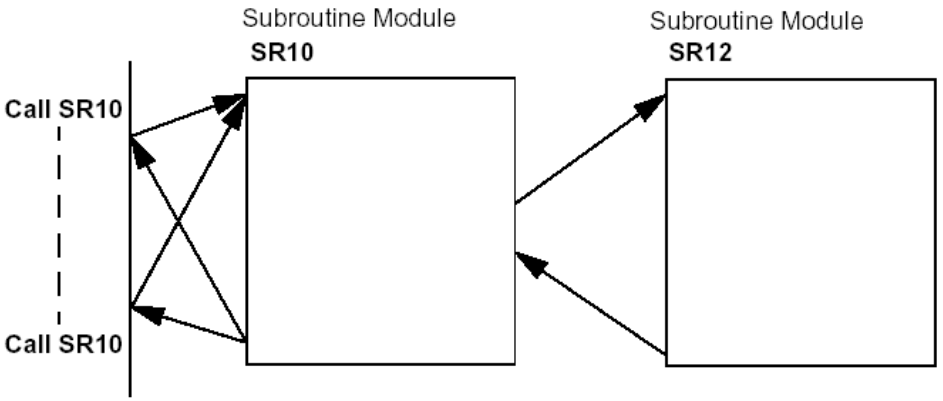
Rank	Instruction
1	Instruction to an operand
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

Structured Text

Structures for Control of Flux

Subroutines

Call and Return



Structured text language

```
IF %M8 THEN
   RETURN;
END_IF;
```

Structured text language

```
IF (%M5>3) THEN
   RETURN;
END_IF;
IF %M8 THEN
   %MD26:=%MW4*%KD6;
END_IF;
```

Structured Text

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)

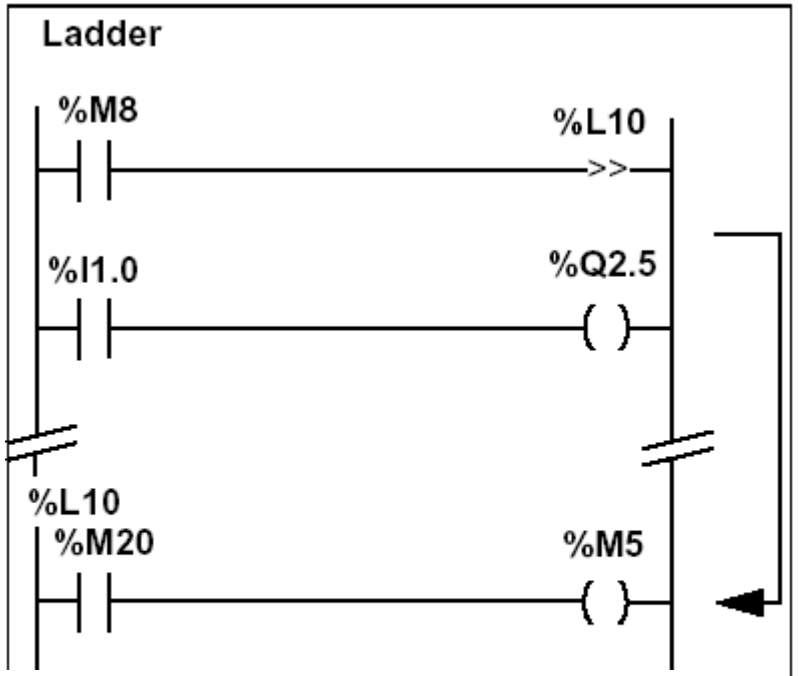
Structured Text

Structures for Control of Flux

Example:

Use of jump instructions

Attention to:




```
IF %M8 THEN
   JUMP %L10;
END_IF;
%Q2.5:=%I1.0;

------
%L10:
  %M5:=%M20;
  %Q2.1:=%I1.0 AND %I1.2;
```

Jump to label %L10
if %M8=1

Structured Text

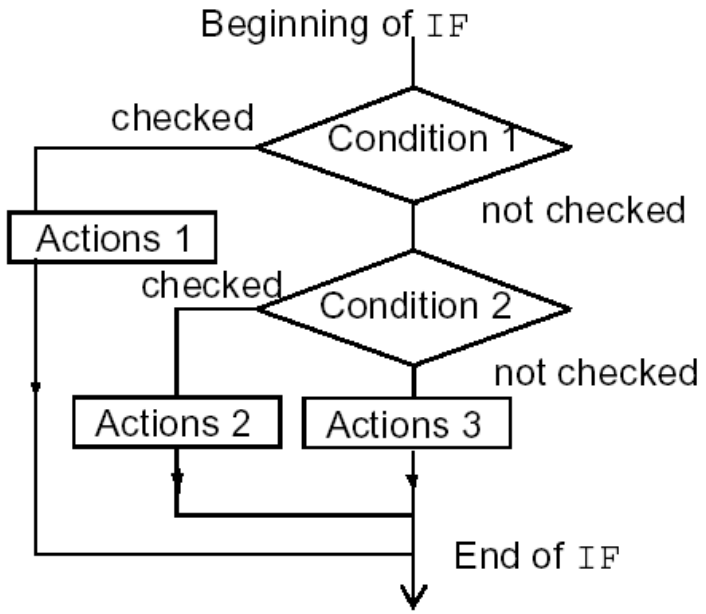
Structures for Control of Flux

IF ... THEN ... ELSE ...

Syntax

```
IF condition1 THEN
    actions1;
ELSEIF condition2 THEN
    actions2;
ELSE
    actions3;
END_IF;
```

Operation



Structured Text

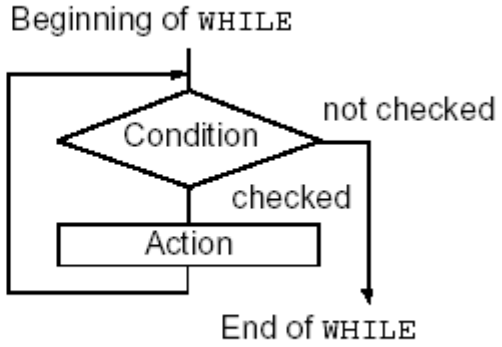
Structures for Control of Flux

WHILE

Syntax

```
WHILE condition DO

    action ;

END_WHILE;
```

Operation



Example:

```
! (*WHILE conditional repeated action*)
  WHILE %MW4<12 DO
       INC %MW4;
       SET %M25[%MW4];
  END_WHILE;
```

Structured Text

Structures for Control of Flux

REPEAT ... UNTIL

FOR ... DO

EXIT to abort the execution of a st6ructured flux control instruction

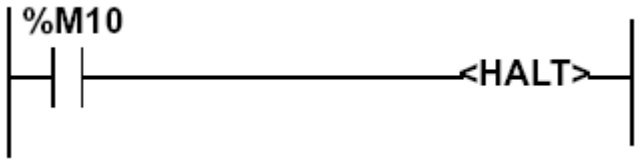
Example:

```
! (*Instruction for exiting the loop EXIT*)
  WHILE %MW1<124 DO
       %MW2:=0;
       %MW3:=%MW100[%MW1];
       REPEAT
             %MW500[%MW2]:=%MW3+%MW500[%MW2];
             IF(%MW500[%MW2]>32700) THEN
                   EXIT;
             END_IF;
             INC %MW2;
       UNTIL %MW2>25 END_REPEAT;
       INC %MW1;
  END_WHILE;
```

Structured Text

Structures for Control of Flux

Halt

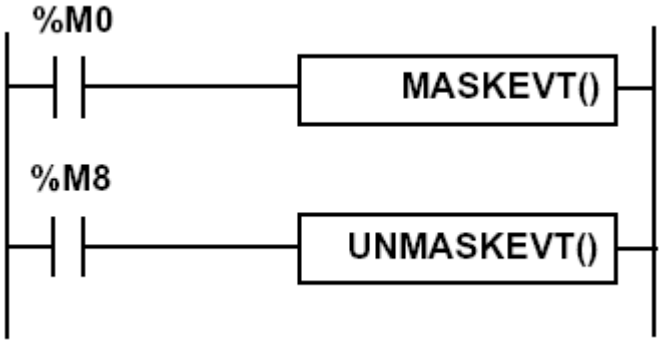


Structured text language

```
IF %M10 THEN
    HALT;
END_IF;
```

Stops all processes!

Events masking



Structured text language

```
IF %M0 THEN
   MASKEVT();
END_IF;
IF %M8 THEN
   UNMASKEVT();
END_IF;
```

Structured Text

Data and time related instructions

Name	Function
SCHEDULE	Time function
RRTC	Reading system date
WRTC	Updating system date
PTC	Reading date and stop code
ADD_TOD	Adding a duration to a time of day
ADD_DT	Adding a duration to a date and time
DELTA_TOD	Measuring the gap between times of day
DELTA_D	Measuring the gap between dates (without time).
DELTA_DT	Measuring the gap between dates (with time).
SUB_TOD	Totaling the time to date
SUB_DT	Totaling the time to date and time
DAY_OF_WEEK	Reading the current day of the week
TRANS_TIME	Converting duration into date
DATE_TO_STRING	Converting a date to a character string
TOD_TO_STRING	Converting a time to a character string
DT_TO_STRING	Converting a whole date to a character string
TIME_TO_STRING	Converting a duration to a character string

Structured Text

There are other advanced instrauctions (see manual)

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

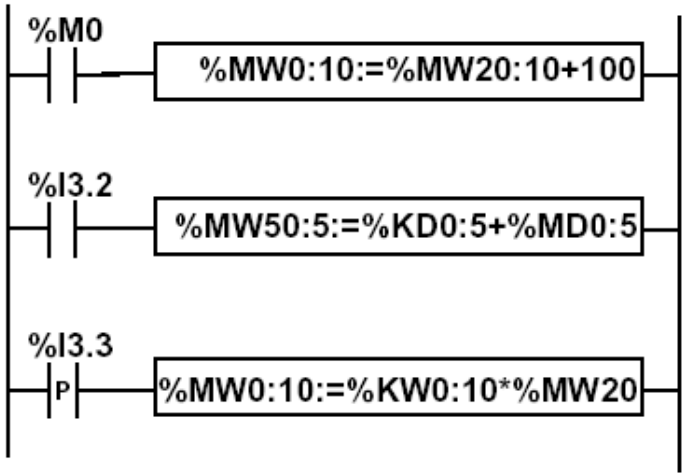
...

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

Structured Text

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]
```