Industrial Automation
(Automação de Processos Industriais)

PLC Programming languages
Ladder Diagram

http://users.isr.ist.utl.pt/~jag/courses/api1314/api1314.html

Syllabus:

Chap. 2 – Introduction to PLCs [2 weeks]

...

 Chap. 3 – PLC Programming languages [2 weeks]
Standard languages (IEC-61131-3):
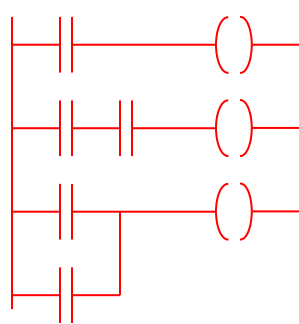*Ladder Diagram; Instruction List,* and *Structured Text.*
Software development resources.

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

PLC Programming languages
(IEC 1131-3 changed to IEC 61131-3)
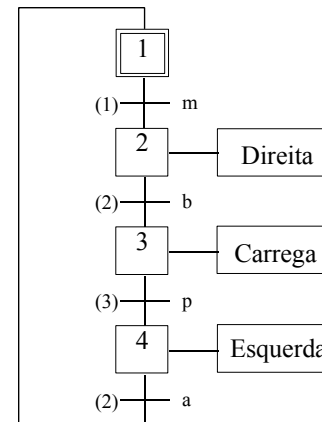
Ladder Diagram

Structured Text

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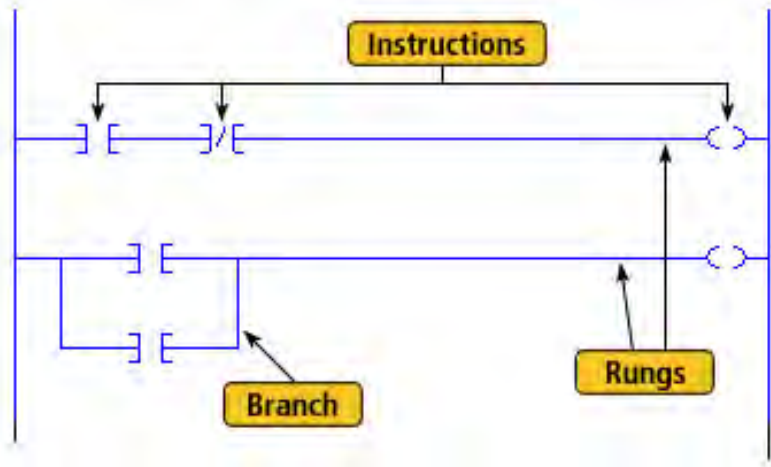
Instruction List

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

Sequential Function Chart
(GRAFCET)
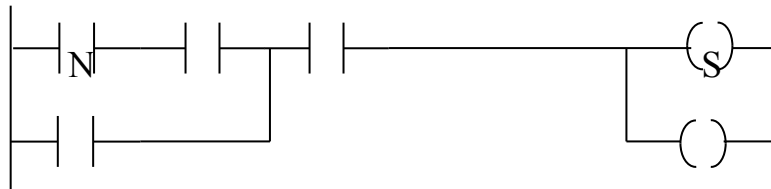
Ladder diagram

A **program** is a series of instructions that directs the PLC to execute actions.

Relay ladder logic, the standard programming language, is based on electromagnetic relay control.

Ladder diagram Types of operands in Schneider DMY 28FK:

CPU objects

System: ☐ %S ☐ %SW   [Select all]

Memory: ☐ %M ☐ %MW ☐ %MD ☐ %MF

    ☐ %KV ☐ %KD ☐ %KF [Unselect all]

I/O Objects

Channel: ☐ %CH

Configuration: ☐ %KV ☐ %KD ☐ %KF [Select all]

System: ☐ %MW

Status: ☐ %MW   [Unselect all]

Parameter: ☐ %MW ☐ %MD ☐ %MF

Command: ☐ %MW ☐ %MD ☐ %MF

Implicits: ☑ %I ☐ %IV ☐ %ID ☐ %IF ☐ %IERR

   ☑ %Q ☐ %QW ☐ %QD ☐ %QF

Update

[Update grid with] ☑ addresses

     ☐ names, types and comments

     ☐ usages

[Filter on usage]

	Address	Name	Type	Comment
1	%I0.3.0		EBOOL	
2	%I0.3.1		EBOOL	
3	%I0.3.2		EBOOL	
4	%I0.3.3		EBOOL	
5	%I0.3.4		EBOOL	
6	%I0.3.5		EBOOL	
7	%I0.3.6		EBOOL	
8	%I0.3.7		EBOOL	
9	%I0.3.8		EBOOL	
10	%I0.3.9		EBOOL	
11	%I0.3.10		EBOOL	
12	%I0.3.11		EBOOL	
13	%I0.3.12		EBOOL	
14	%I0.3.13		EBOOL	
15	%I0.3.14		EBOOL	
16	%I0.3.15		EBOOL	
17	%Q0.3.16		EBOOL	
18	%Q0.3.17		EBOOL	
19	%Q0.3.18		EBOOL	
20	%Q0.3.19		EBOOL	
21	%Q0.3.20		EBOOL	
22	%Q0.3.21		EBOOL	
23	%Q0.3.22		EBOOL	
24	%Q0.3.23		EBOOL	
25	%Q0.3.24		EBOOL	
26	%Q0.3.25		EBOOL	
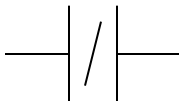
Ladder diagram   Types of operands:
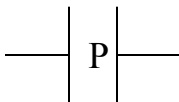
| Bits	Description	Examples	Write access
Immediate values	0 or 1 (False or True)	0	–
Inputs/outputs	These bits are the "logic images" of the electrical states of the inputs/outputs. They are stored in the data memory and updated each time the task in which they are configured is polled. **Note:** The unused input/output bits may not be used as internal bits.	%I23.5 %Q51,2	No Yes
Internal	The internal bits are used to store the intermediary states during execution of the program.	%M200	Yes
System	The system bits %S0 to %S127 monitor the correct operation of the PLC and the running of the application program.	%S10	According to i
Function blocks	The function block bits correspond to the outputs of the function blocks or DFB instance. These outputs may be either directly connected or used as an object.	%TM8.Q	No
Word extracts	With the PL7 software it is possible to extract one of the 16 bits of a word object.	%MW10:X5	According to the type of words
Grafcet steps and macro-steps	The Grafcet status bits of the steps, macro-steps and macro-step steps are used to recognize the Grafcet status of step i, of macro-step j or of step i of the macro-step j.	%X21 %X5.9	Yes Yes

Ladder diagram

Basic Instructions

Load

Normally open contact: contact is active (result is 1) when the control bit is 1.

Normally closed contact: contact is active (result is 1) when 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.

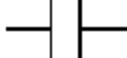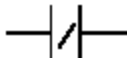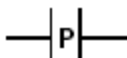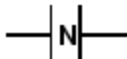Contact in the **falling edge**: contact is active during a scan cycle where the control bit has a falling edge.
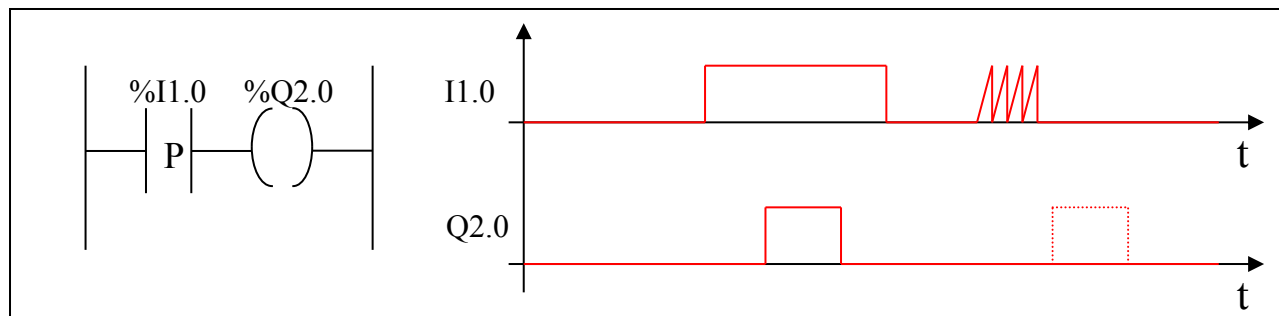
Ladder diagram

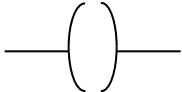## Basic Instructions

Load operands

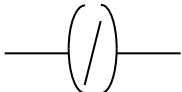**Permitted operands**   The following table gives a list of the operands used for these instructions.

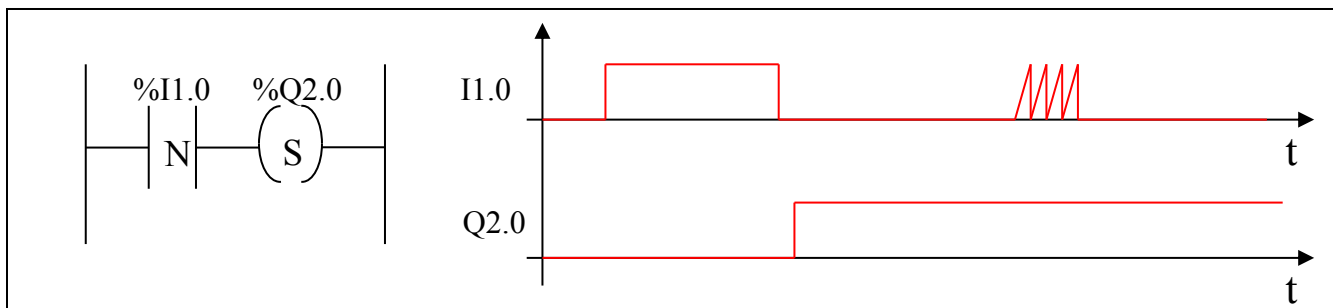| Ladder	Instruction list	Structured text	Operands
—│├—	LD	:=	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi, (True and False in instruction list or structured text)
—│/├—	LDN	:=NOT	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi, (True and False in instruction list or structured text)
—│P├—	LDR	:=RE	%I,%Q,%M
—│N├—	LDF	:=FE	%I,%Q,%M

Ladder diagram

Basic Instructions

Store   —( )—   The result of the logic function activates the coil.

—(/)—   The inverse result of the logic function activates the coil.

—(S)—   The result of the logic function energizes the relay
(sets the latch).

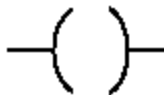—(R)—   The result of the logic function de-energizes the relay
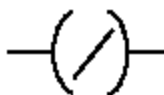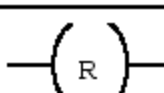(resets the latch)..

Ladder diagram

Basic Instructions

Store operands

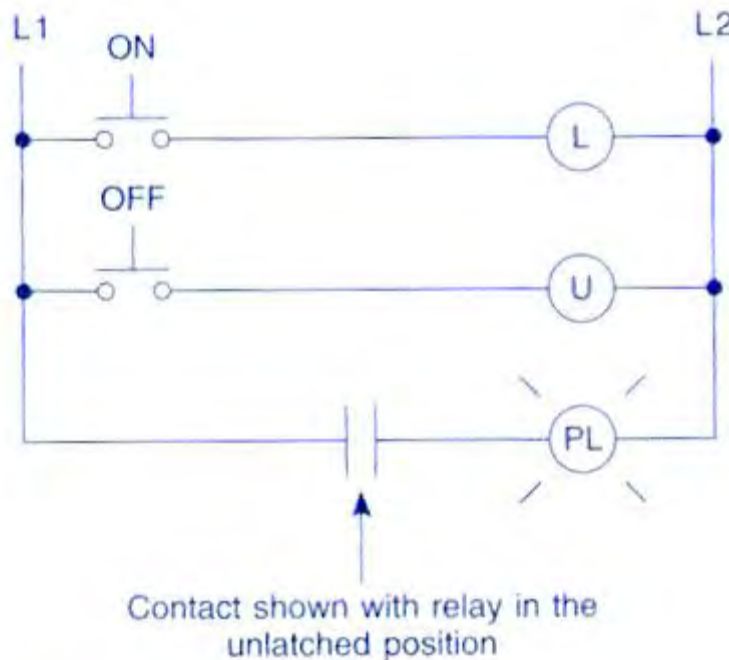**Permitted operands**   The following table gives a list of the operands used for these instructions

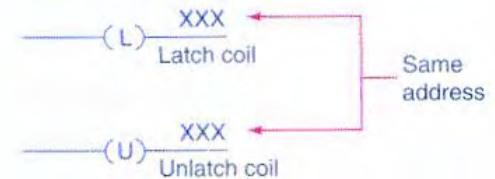| Language data	Instruction list	Structured text	Operands
—( )—	ST	:=	%I,%Q,%M,%S,%•:Xk
—(/)—	STN	:=NOT	%I,%Q,%M,%S,%•:Xk
—(S)—	S	SET	%I,%Q,%M,%S,%•:Xk,%Xi Only in the preliminary processing.
—(R)—	R	RESET	%I,%Q,%M,%S,%•:Xk,%Xi Only in the preliminary processing.

Ladder diagram

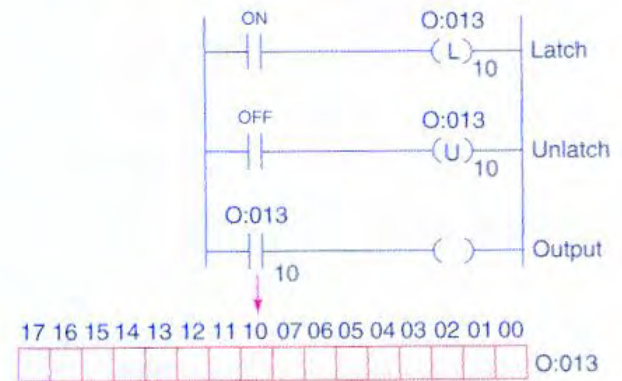## *Allen Bradley notation*

Relays with *latch* and *unlatch*

Instruction	Symbol	Mnemonic
Output latch	—(L)—	OTL
Output unlatch	—(U)—	OTU

XXX
—(L)— Latch coil
XXX
—(U)— Unlatch coil

Same address

(a) Latch and unlatch coils have the same address

ON   O:013
—| |—   —(L)— Latch
                      10

OFF   O:013
—| |—   —(U)— Unlatch
                      10

O:013
—| |—   —( )— Output
   10

17 16 15 14 13 12 11 10 07 06 05 04 03 02 01 00

O:013

(b) Control logic

Fig. 6-50
Schematic of electromagnetic latching relay.
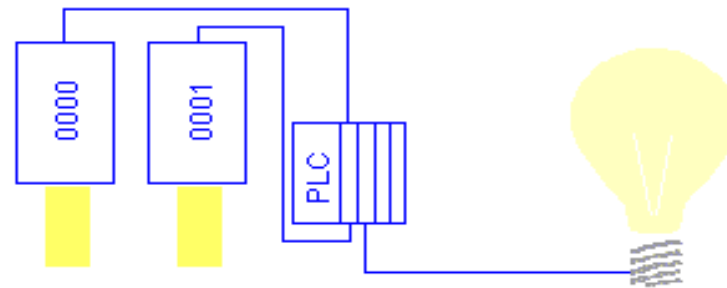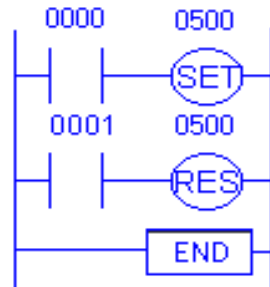
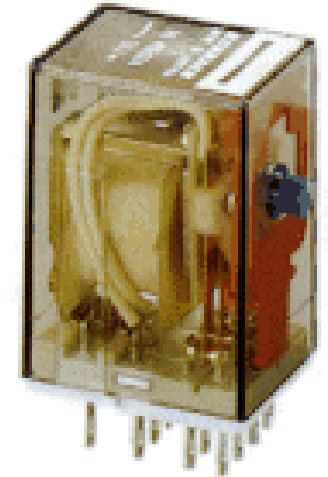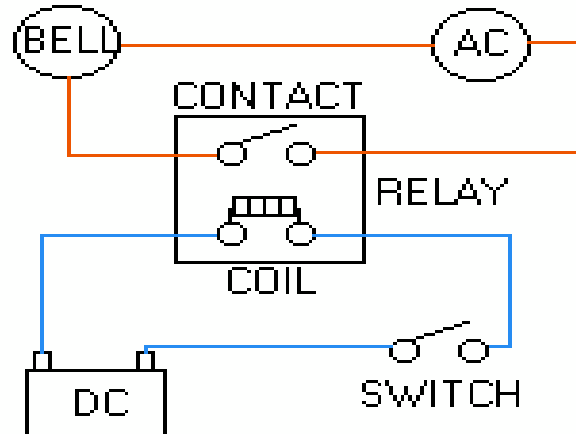Contact shown with relay in the unlatched position

Fig. 6-51
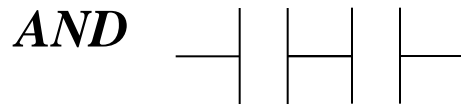OUTPUT LATCH and OUTPUT UNLATCH instructions.

Ladder diagram

Relay-type instructions

Example:

BELL — CONTACT — AC

CONTACT

RELAY

COIL

DC — SWITCH

```
0000    0500
 | |    (SET)

0001    0500
 | |    (RES)

        [END]
```

0000   0001   PLC

Ladder diagram

Basic Instructions

AND   —| |—   AND of the operand with the result of the previous logical operation.

—| / |—   AND of the operand with the inverted result of the previous logical operation.

—| P |—   AND of the rising edge with the result of the previous logical operation.

—| N |—   AND of the falling edge with the result of the previous logical operation.

%I1.0   %I1.0   %Q2.0
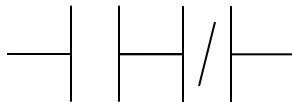
—| N |——| P |——(S)—

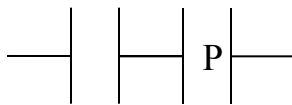I1.0

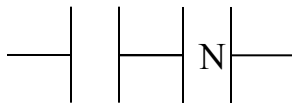Q2.0

t

t

Ladder diagram

Basic Instructions

OR

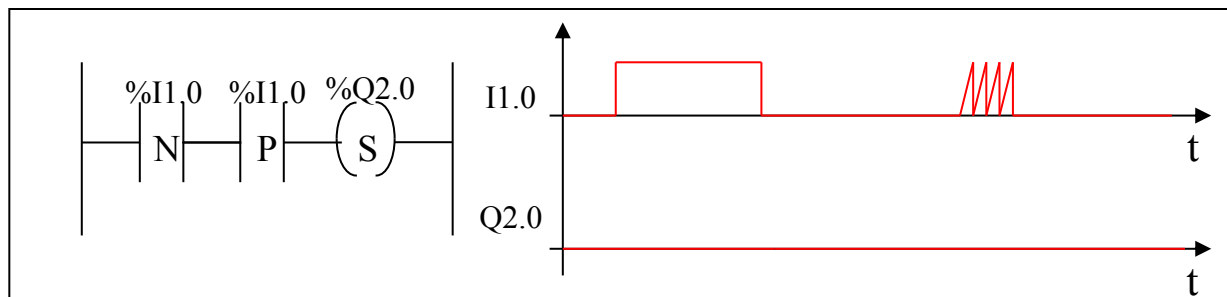OR of the operand with the result of the previous logical operation.

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

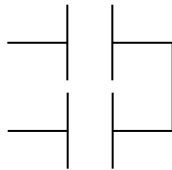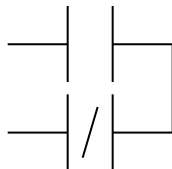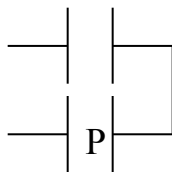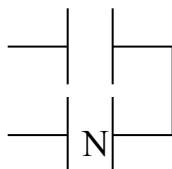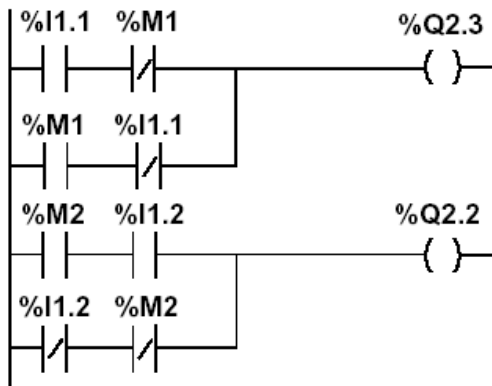OR of the rising edge with the result of the previous logical operation.

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

Ladder diagram

Basic Instructions

XOR

Instruction list	Structured text	Description	Timing diagram
XOR	XOR	OR Exclusive between the operand and the previous instruction's Boolean result	XOR %I1.1 %M1 %Q2.3
XORN	XOR (NOT...)	OR Exclusive between the operand inverse and the previous instruction's Boolean result	XORN %M2 %I1.2 %Q2.2
XORR	XOR (RE...)	OR Exclusive between the operand's rising edge and the previous instruction's Boolean result	XORR %I1.3 %I1.4 %Q2.4
XORF	XOR (FE...)	OR Exclusive between the operand's falling edge and the previous instruction's Boolean result.	XORF %M3 %I1.5 %Q2.5

%Q2.3 := %I1.1 XOR %M1;
%Q2.2 := NOT(%M2 XOR %I1.2);
%Q2.2 := %M2 XOR NOT(%I1.2);

Ladder diagram

Ladder assembling
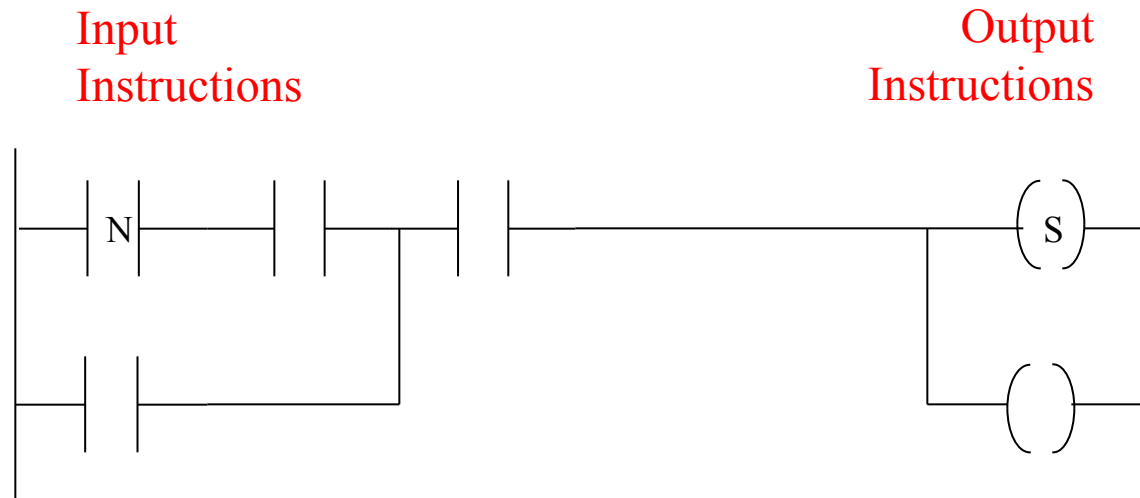
The outputs that have a TRUE logical function, evaluated from the left to right and from the top to the bottom, are energized (Schneider, Micro PLCs).

Ladder diagram

Example:
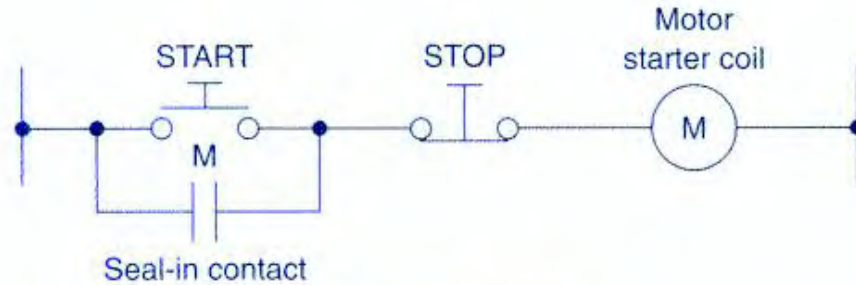
(a) Hard-wired circuit
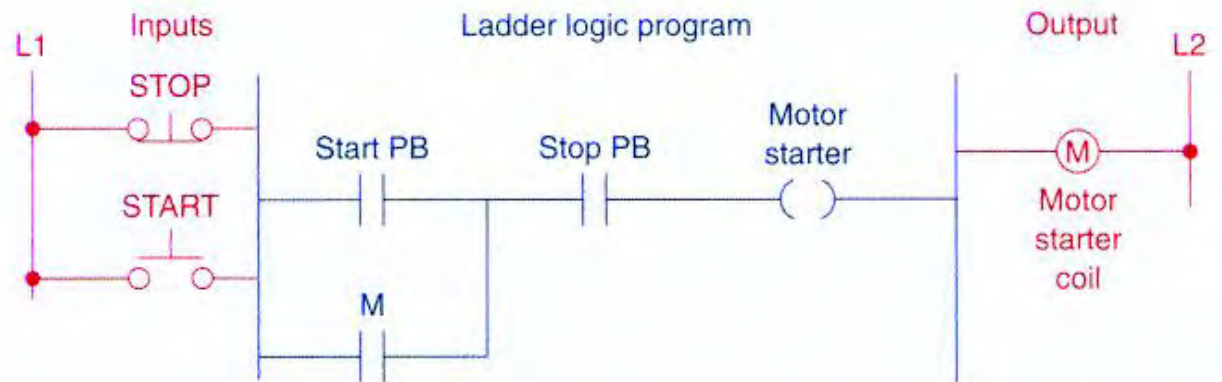
(b) Programmed circuit

Fig. 6-48

Seal-in circuit.

Ladder diagram

Example:
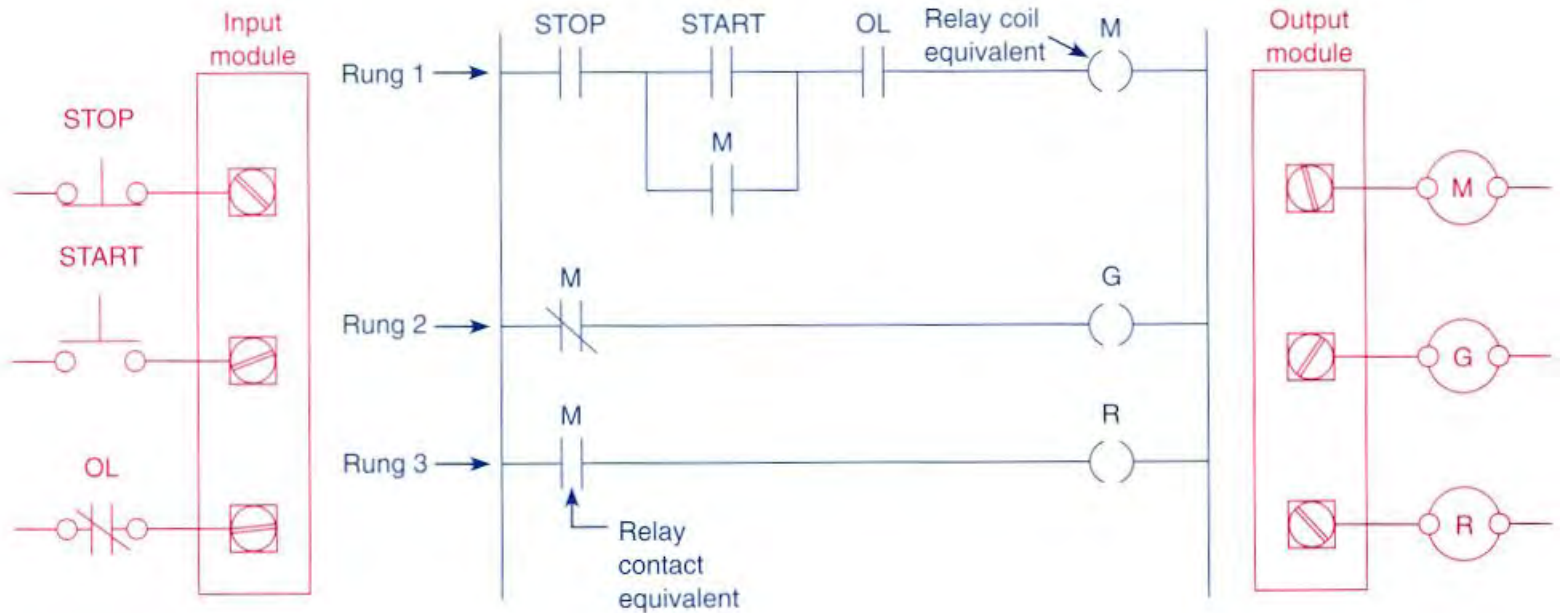
Fig. 4-24

Ladder logic program.

Ladder diagram
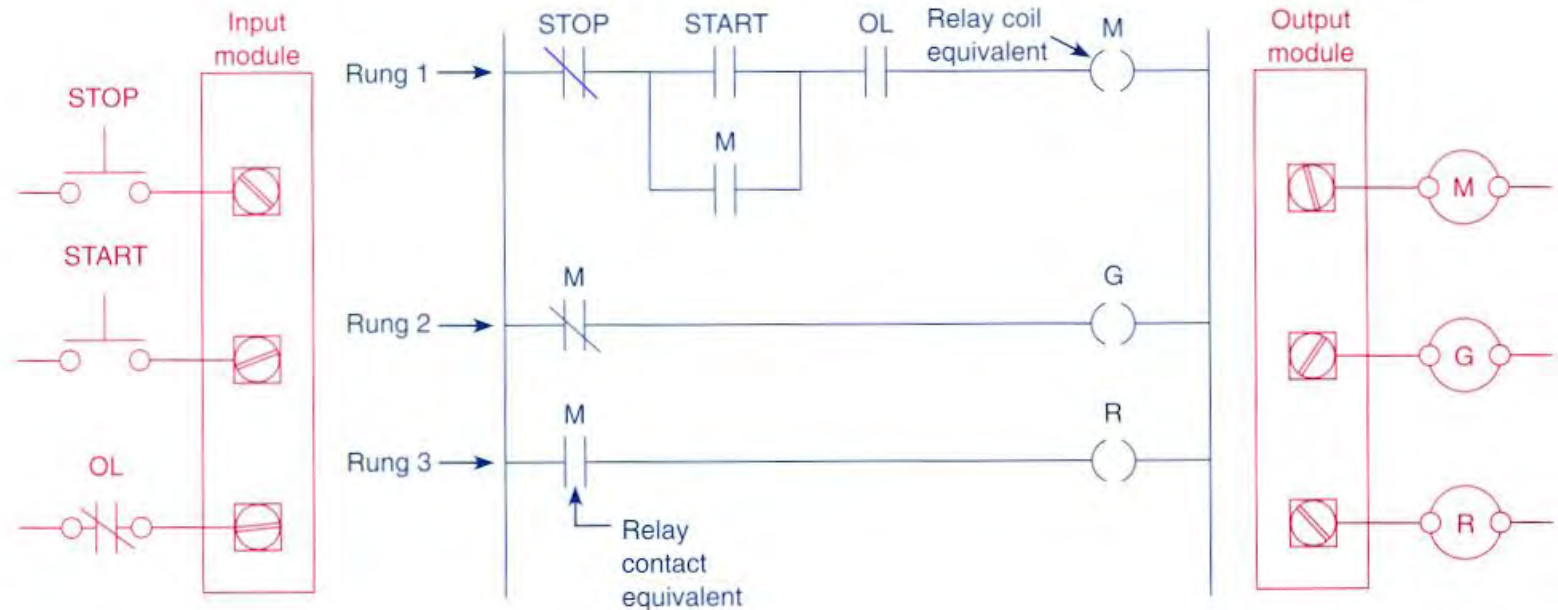
Example:

If the Stop PB is normally open, then invert its logic in the ladder diagram.

Ladder diagram

Example:

Relay schematic

Ladder logic program

Gate logic
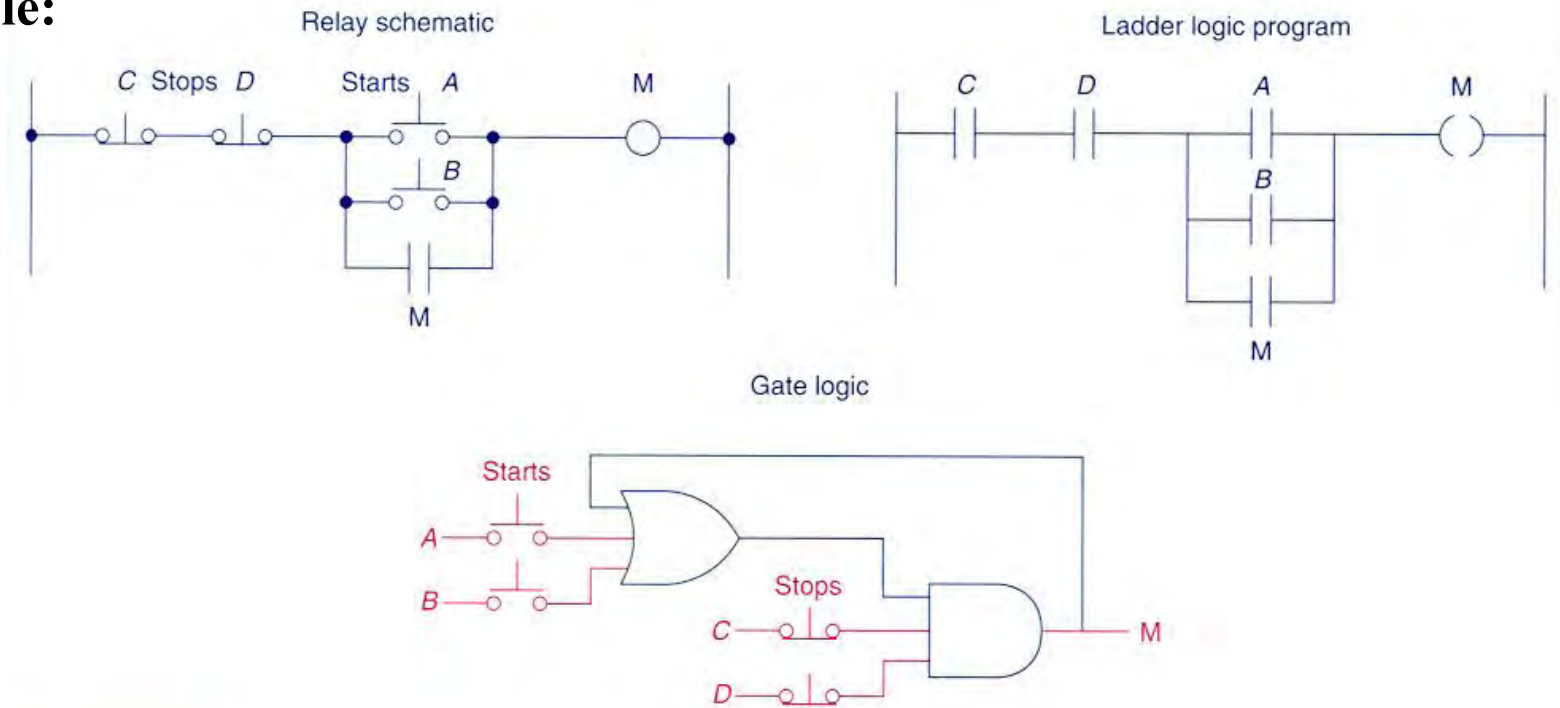
Example 4-9

A motor control circuit with two stop buttons. When the start button is depressed, the motor runs. By sealing, it continues to run when the start button is released. The stop buttons stop the motor when they are depressed.

Ladder diagram

General case of Inputs and Outputs in **parallel**, with **derivations**
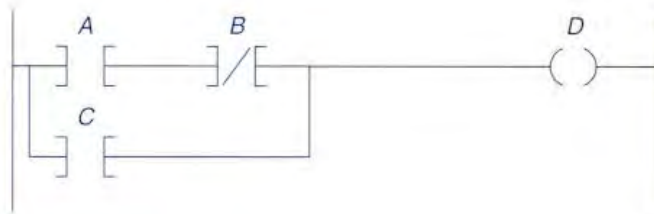
Fig. 5-21
Parallel input branching.
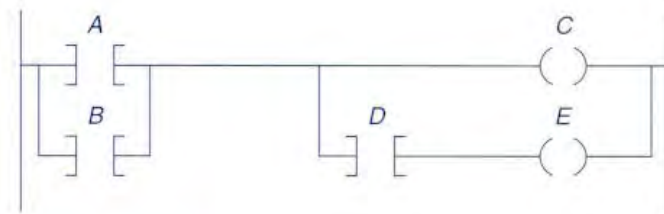
Fig. 5-23
Parallel output branching with conditions.

Fig. 5-22
Parallel output branching.
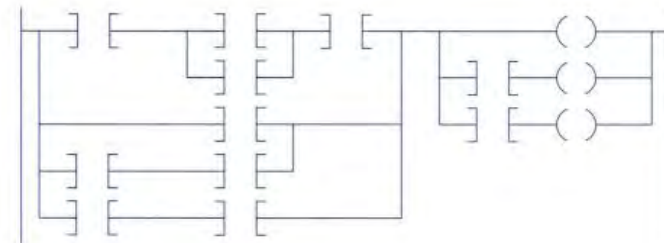
Fig. 5-24
Nested input and output branches.

Note: it is important to study the **constraints** and **potentialities** of the development tools.

Ladder diagram

Imbricated (nested) contacts and **alternative** solution
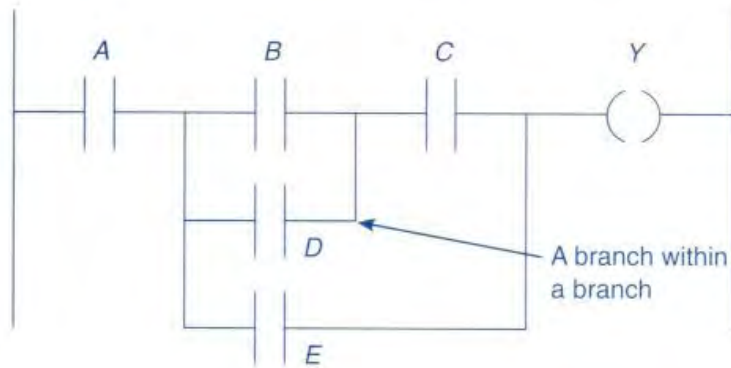
Fig. 5-25
Nested contact program.
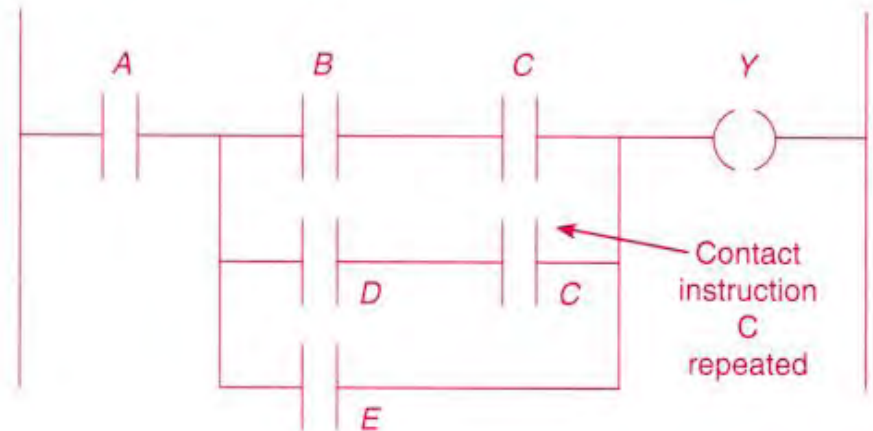
Fig. 5-26
Program required to eliminate nested contact.

Ladder diagram

Contacts in the **vertical** and **alternative** solution
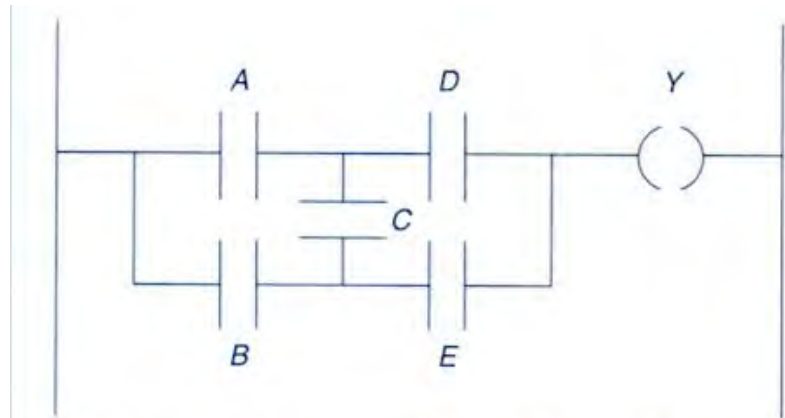
Boolean equation: $Y = (AD) + (BCD) + (BE) + (ACE)$

Fig. 5-28

Program with vertical contact
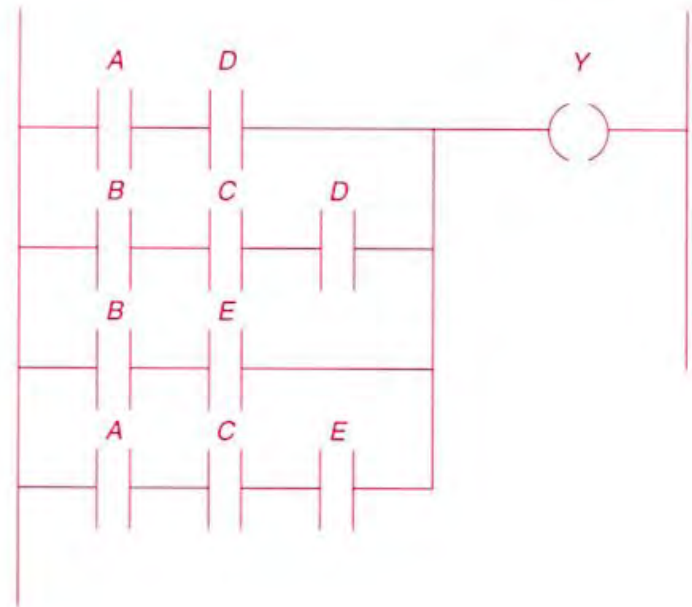
Fig. 5-29

Reprogrammed to eliminate vertical contact.

Ladder diagram

Contacts in the **vertical** and **alternative** solution

Another example:
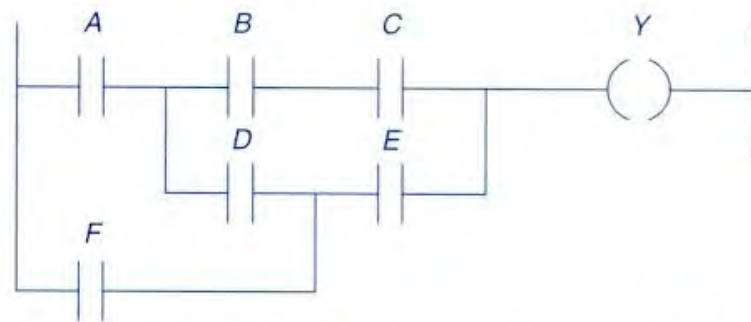
Boolean equation: $Y = (ABC) + (ADE) + (FE) + (FDBC)$

Fig. 5-30
Original circuit.
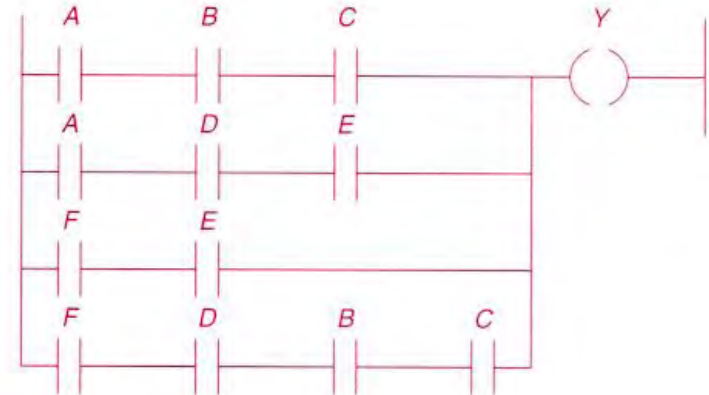
Fig. 5-31
Reprogrammed circuit.

Solves the problem of disallowed right to left scanning (FDBC in fig5.30).

Ladder diagram　*Temporized Relays or Timers*

Solid-state timing relay　　Pneumatic timing relay　　Plug-in timing relay

Ladder diagram   *Temporized Relays or Timers (pneumatic)*

Pneumatic timing relay

Operating coil

Normally open terminals

Instantaneous contacts

Normally closed terminals

Time control contacts

Normally open terminals

Normally closed terminals

Time adjustment
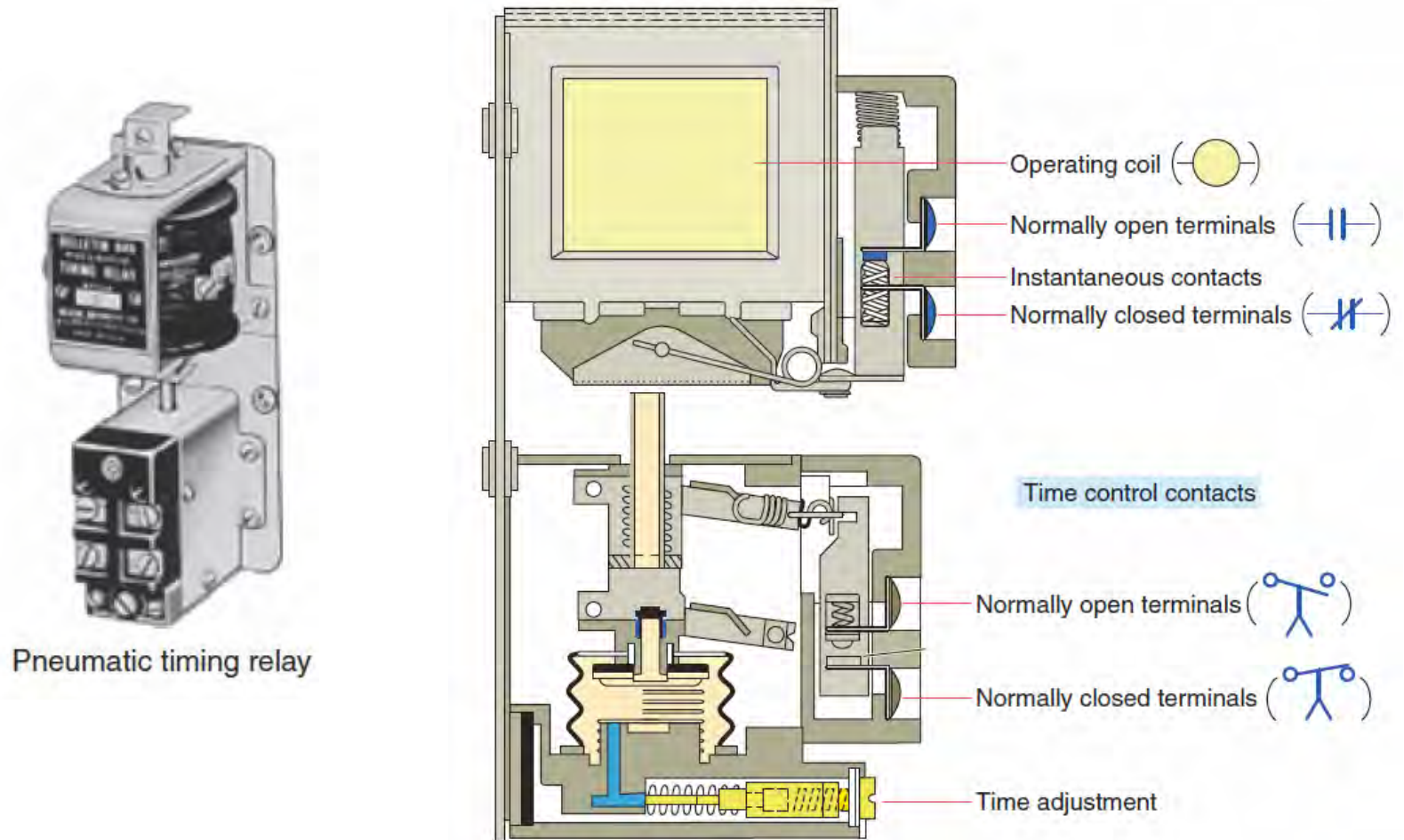
The **instantaneous** contacts change state as soon as the timer coil is powered.
The **delayed** contacts change state at the end of the time delay.

Ladder diagram   *Temporized Relays or Timers*

On-delay, provides time delay when the relay coil is energized.

Off-delay, provides time delay when the relay coil is de-energized.

On-delay symbols	
Normally open, timed closed contact (NOTC).	Normally closed, timed open contact (NCTO).
Contact is open when relay coil is de-energized.	Contact is closed when relay coil is de-energized.
When relay is energized, there is a time delay in closing.	When relay is energized, there is a time delay in opening.

Off-delay symbols	
Normally open, timed open contact (NOTO).	Normally closed, timed closed contact (NCTC).
Contact is normally open when relay coil is de-energized.	Contact is normally closed when relay coil is de-energized.
When relay coil is energized, contact closes instantly.	When relay coil is energized, contact opens instantly.
When relay coil is de-energized, there is a time delay before the contact opens.	When relay coil is de-energized, there is a time delay before the contact closes.

Tables: Relay *symbols* used for timed contacts.

Ladder diagram   *Temporized Relays or Timers*
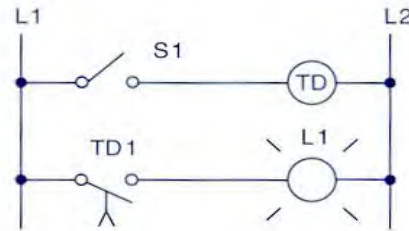
Sequence of operation:
S1 open, TD de-energized, TD1 open, L1 off.

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.

(a)
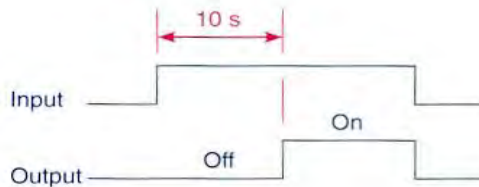
(b)

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.

(a)
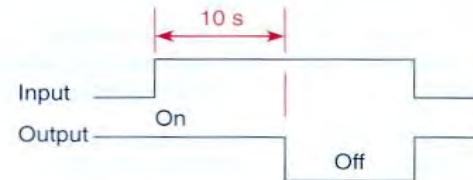
(b)

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

Ladder diagram   *Temporized Relays or Timers*

Sequence of operation:
S1 open, TD de-energized, TD1 open, L1 off.

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

S1 is opened, TD de-energizes, timing period starts,
TD1 is still closed, L1 is still on.

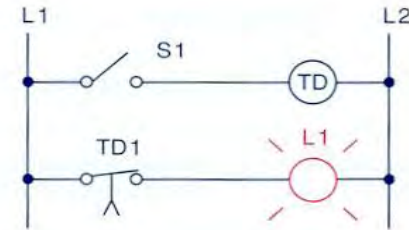After 10 s, TD1 opens, L1 is switched off.

(a)
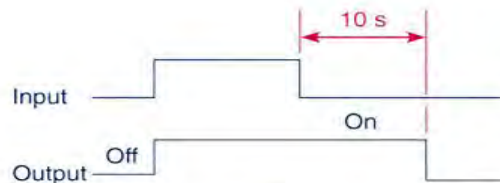
(b)

Fig. 7-5
Off-delay timer circuit (NOTO contact). (a) Operation.
(b) Timing diagram.
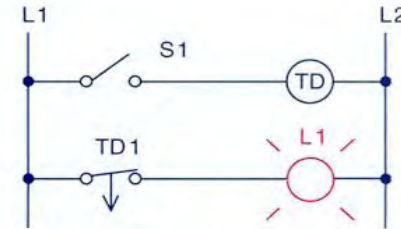
Sequence of operation:
S1 open, TD de-energized, TD1 closed, L1 on.

S1 closes, TD energizes, TD1 opens instantly,
L1 is switched off.

S1 is opened, TD de-energizes, timing period starts,
TD1 is still open, L1 is still off.
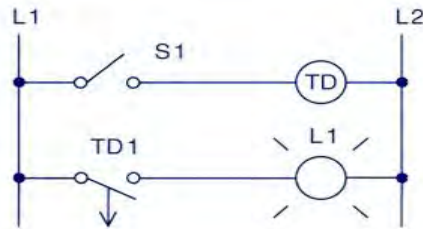
After 10 s, TD1 closes, L1 is switched on.

(a)
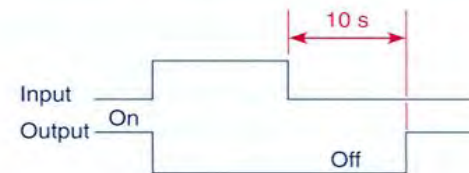
(b)

Fig. 7-6
Off-delay timer circuit (NCTC contact). (a) Operation.
(b) Timing diagram.

Ladder diagram

Temporized Relays

or Timers (PLC)

%TMi

```
 ┌─────────────────┐
─┤ IN            Q ├─
 │                 │
 │                 │
 │  MODE: TON      │
 │  TB: 1mn        │
 │                 │
 │  TM.P: 9999     │
 │  MODIF: Y       │
 └─────────────────┘
```
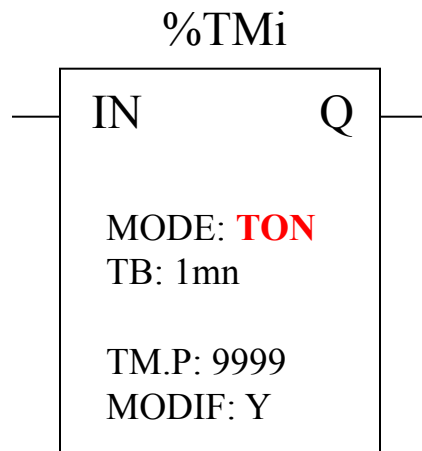
Characteristics:

Identifier: %TMi		0..63 in the TSX37
Input:	IN	to activate
Mode:	**TON**	**Timer On delay**
	TOF	**Timer 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 read or tested)
Modifiable:	Y/N	can be modified from the console

Ladder diagram

Temporized Relays

or Timers (PLC)

TON mode

%TMi

```
 ──┤ IN          Q ├──
   │                │
   │  MODE: TON     │
   │  TB: 1mn       │
   │                │
   │  TM.P: 9999    │
   │  MODIF: Y      │
```
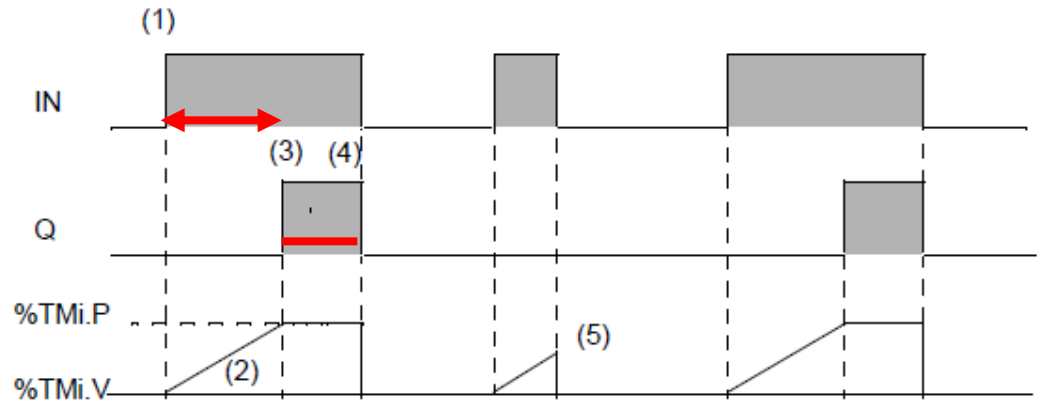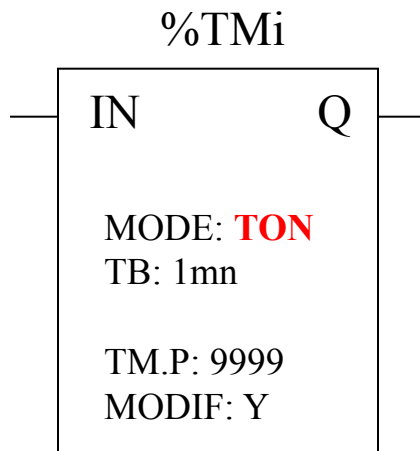
Phase	Description
1	The timer is started with a rising edge on the IN input
2	The current value %TMi.V of the timer increases from 0 to %TMi.P by one unit at each pulse of the time base TB
3	The %TMi.Q output bit moves to 1 when the current value has reached %TMi.P
4	The %TMi.Q output bit stays at 1 while the IN input is at 1.
5	When the IN input is at 0, the timer is stopped even if it is still running: %TMi.V takes the value 0.

App. example: start ringing the alarm if N sec after door open there is no disarm of the alarm.

Page 31

Ladder diagram

TOF mode

Temporized Relays

or Timers (PLC)

%TMi

```
┌──────────────────┐
─┤ IN            Q ├─
│                  │
│                  │
│  MODE: TOF       │
│  TB: 1mn         │
│                  │
│  TM.P: 9999      │
│  MODIF: Y        │
└──────────────────┘
```
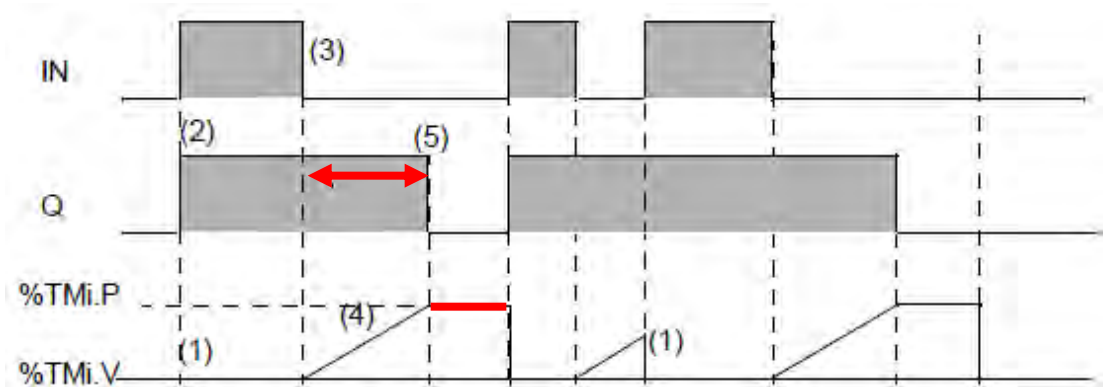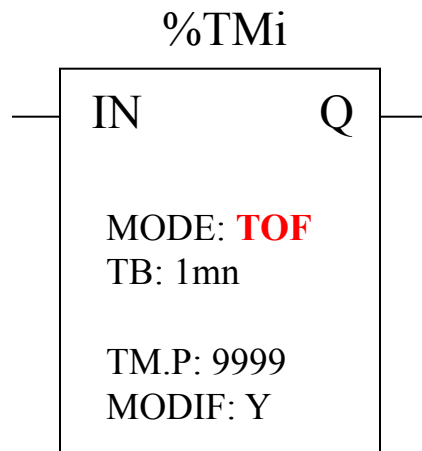
Phase	Description
1	The current value %TMi.V takes 0, on a rising edge of the IN input (even if the timer is running)
2	The %TMi.Q output bit moves to 1.
3	The timer is started with a falling edge on the IN input.
4	The current value %TMi.P increases to %TMi.P by one unit at each pulse of the time base TB.
5	The %TMi.Q output bit returns to 0 when the current value has reached %TMi.P

App. example: turn off stairways lights after N sec the lights' button has been released.

Ladder diagram   *TP mode*

Temporized Relays

or Timers (PLC)

Works as a monostable or as a pulse generator (with pre-programmed period)

%TMi

```
   ┌──────────────┐
───┤ IN        Q  ├───
   │              │
   │  MODE: TP    │
   │  TB: 100msec │
   │              │
   │  TM.P: 5     │
   │  MODIF: Y    │
   └──────────────┘
```
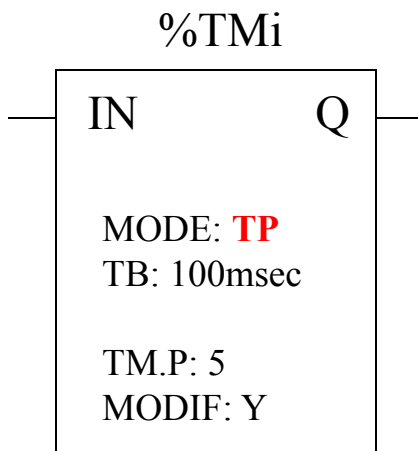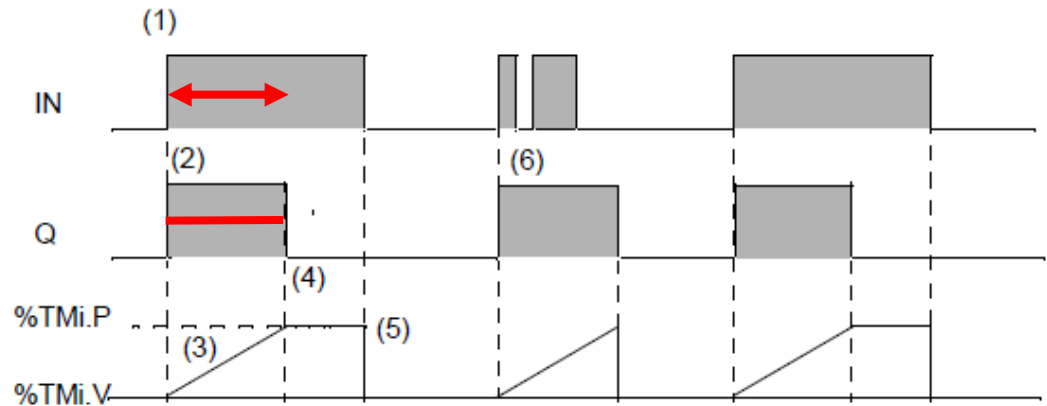
Phase	Description
1	The timer is started with a rising edge on the IN input
2	The %TMi.Q output bit moves to 1.
3	The current value %TMi.V of the timer increases from 0 to %TMi.P by one unit at each pulse of the time base TB
4	The %TMi.Q output bit returns to 0 when the current value has reached %TMi.P.
5	When the IN input and the %TMi.Q output are at 0, %TMi.V takes the value 0.
6	This monostable cannot be reactivated.

App. example: positive input edge give a controlled (fixed) duration pulse to start a motor.

Ladder diagram

Timers in PL7 vs Unity (Schneider)

%TMi

```
┌─────────────────┐
│  IN          Q  │
│                 │
│  MODE: TON      │
│  TB: 1mn        │
│                 │
│  TM.P: 9999     │
│  MODIF: Y       │
└─────────────────┘
```

my_timer1

```
        ┌─────────────┐
        │    TON      │
        │             │
    ────┤ EN     ENO  │
%I0.2.1 │             │         %Q0.4.1
    ────┤ IN       Q  ├──────────( )────
        │             │
  t#5s──┤ PT      ET  │
        └─────────────┘
```
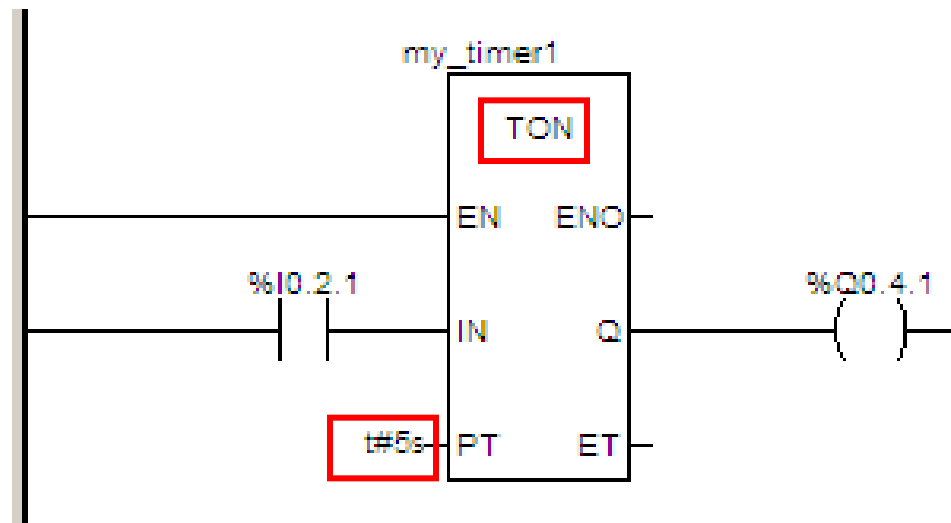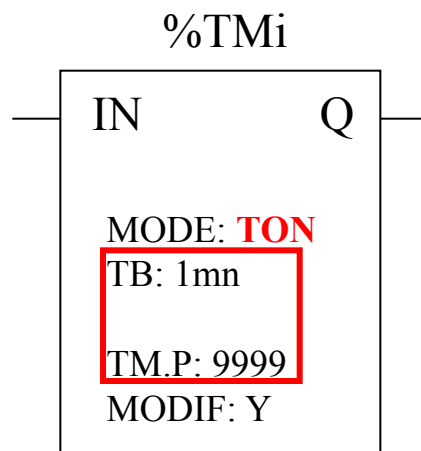
Ladder diagram

Timers in the *Allen-Bradley* PLC-5
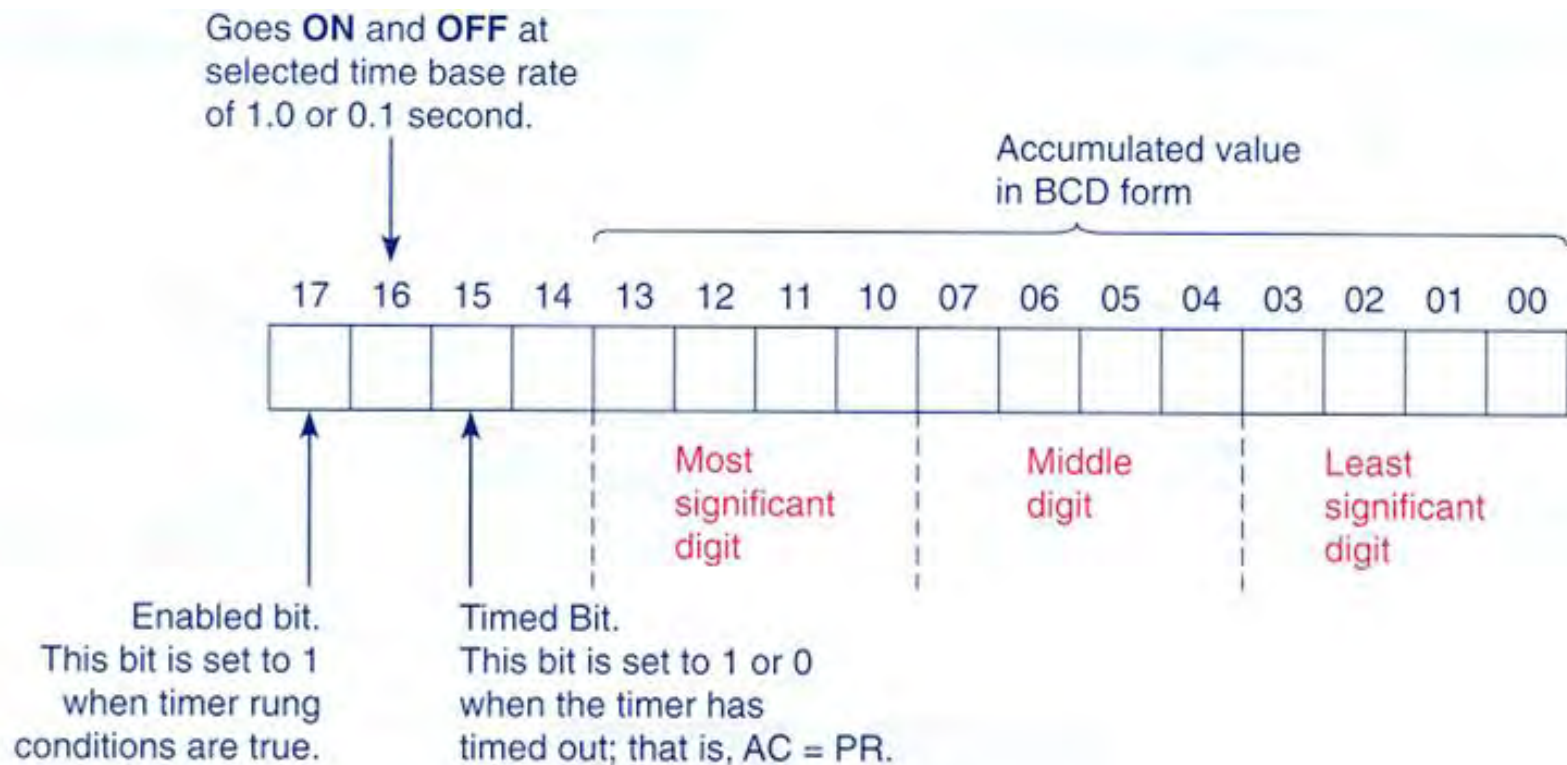
Two alternative representations
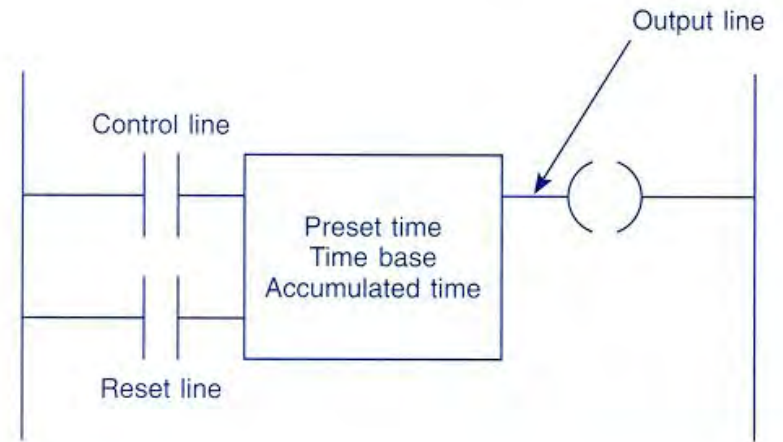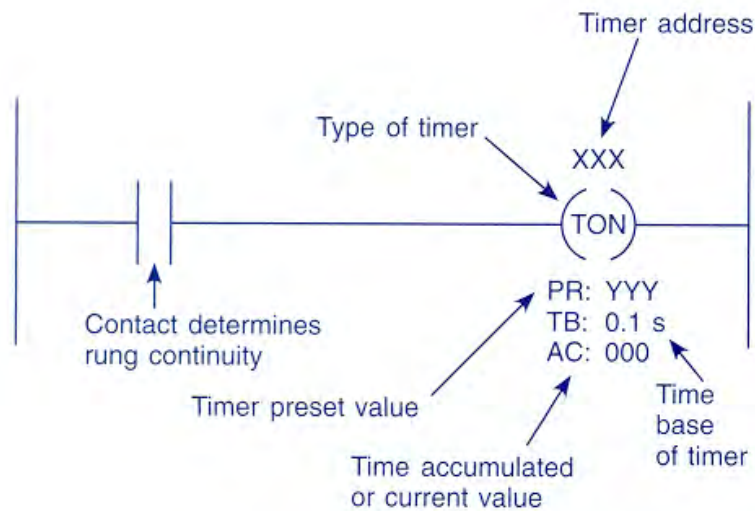
Fig. 7-8
Block-formatted timer instruction.

Ladder diagram

Timers implementation in the *Allen-Bradley* PLC-5:

Goes **ON** and **OFF** at selected time base rate of 1.0 or 0.1 second.

Accumulated value in BCD form

17	16	15	14	13	12	11	10	07	06	05	04	03	02	01	00

Most significant digit

Middle digit

Least significant digit

Enabled bit. This bit is set to 1 when timer rung conditions are true.

Timed Bit. This bit is set to 1 or 0 when the timer has timed out; that is, AC = PR.

Ladder diagram

Retentive Timers

When reset switch is closed, timed bit is reset. Accumulated value is reset and held at zero until reset switch is opened.

Enable bit is reset when input switch is opened.

Input switch 113-06 — True / False

Enable bit 052-17 — On / Off

AC = PR

Preset value

AC value retained when rung condition goes false

Accumulated value

Timed bit 052-15 — On / Off

Output lamp 011-04 — On / Off

Reset switch 113-07 — On / Off

0 1 2 3 4 5 6 7 8 9 10 11 12

Time in seconds

(*b*) Timing chart

Cam-operated contact

Switch-off region

Switch-on region

Motor-driven cam

Motor accumulated motion (rotation) defines the on/off timing.

Fig. 7-20

Electromechanical retentive timer.

Ladder logic program

L1   Inputs   Output   L2

PB1   11306   113   052   01004
Time   06   (RTO)
                            PR: 009
                            TB: 1 s

PB2   11307   Same address
Reset

052   010
15   04
(Timed bit)

113   052
07   (RTR)

(*a*) Programmed logic

Page 48
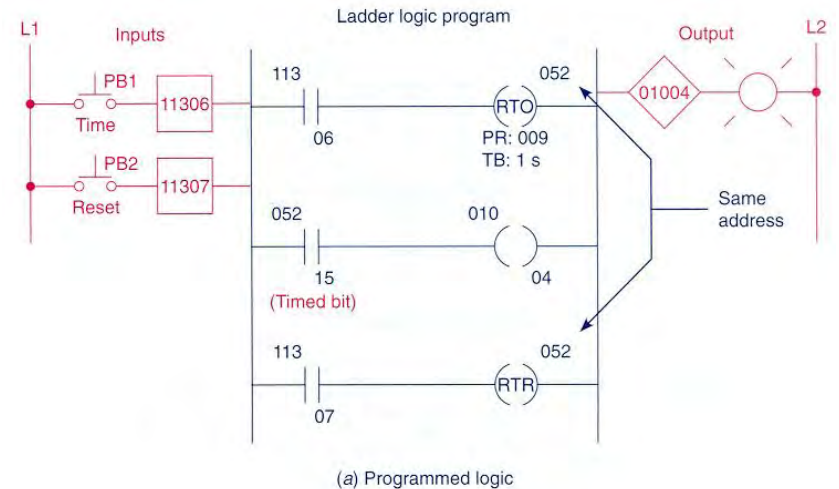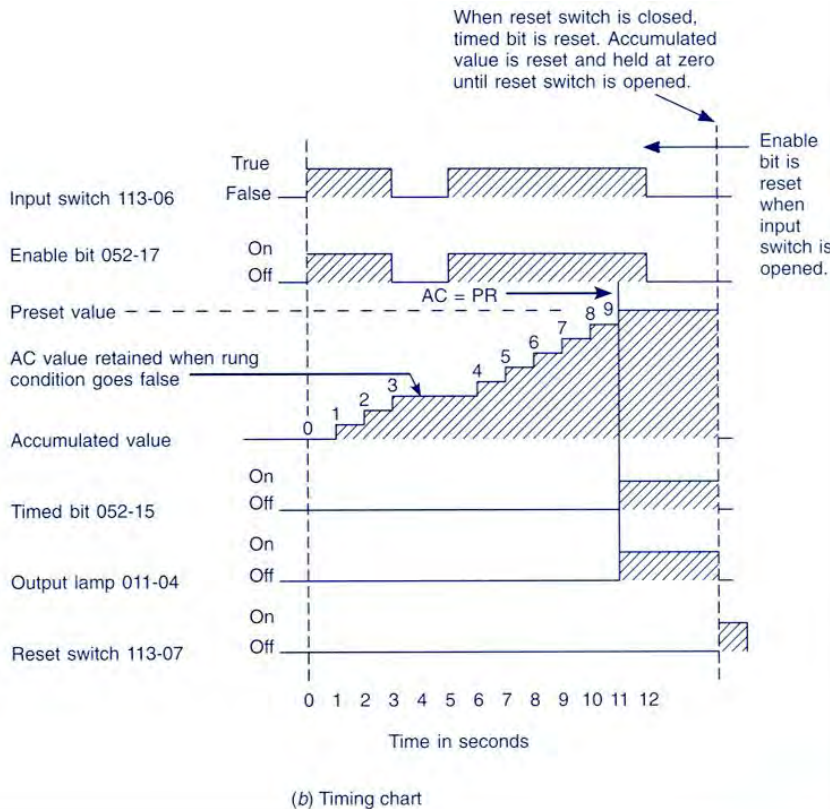
Ladder diagram
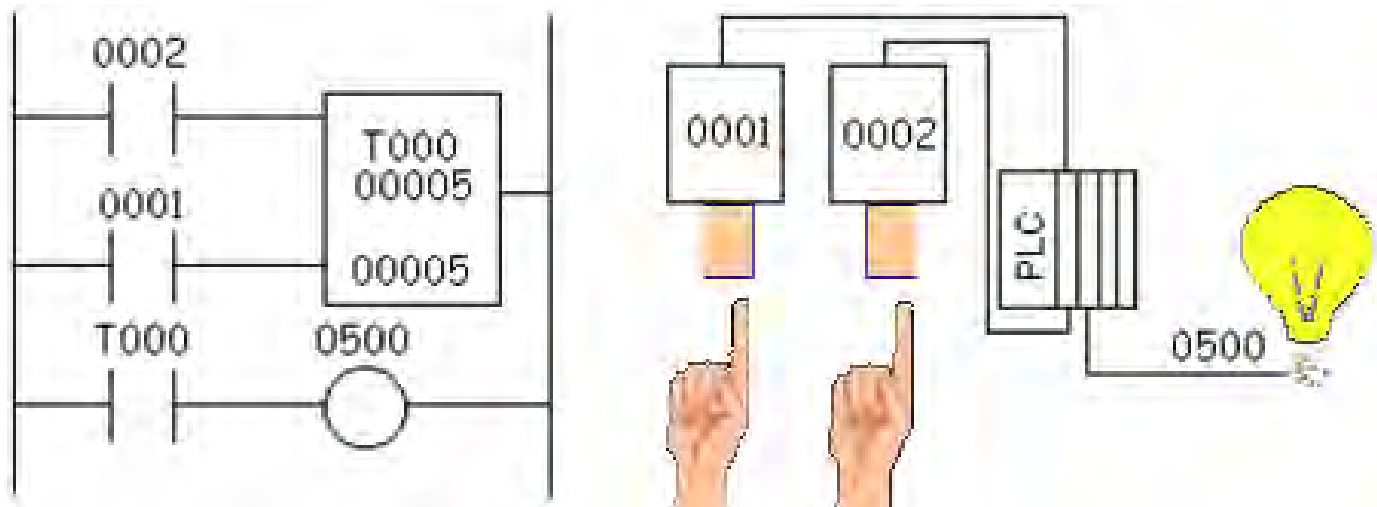
Example of *retentive timers*
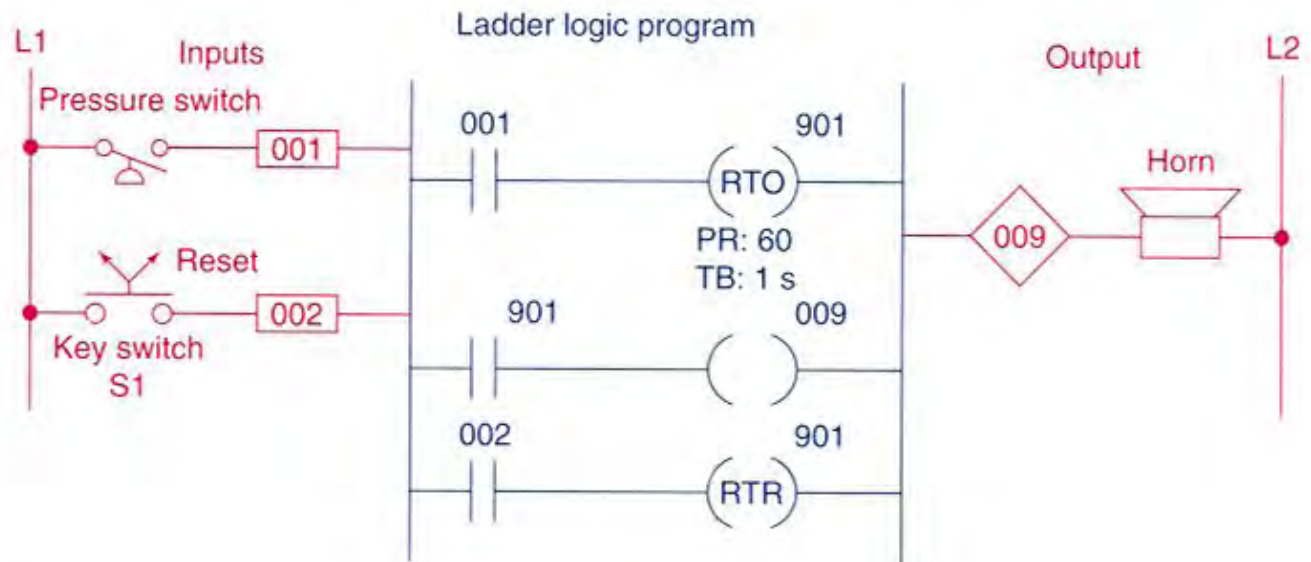
Fig. 7-22

Retentive on-delay alarm program.

Ladder diagram
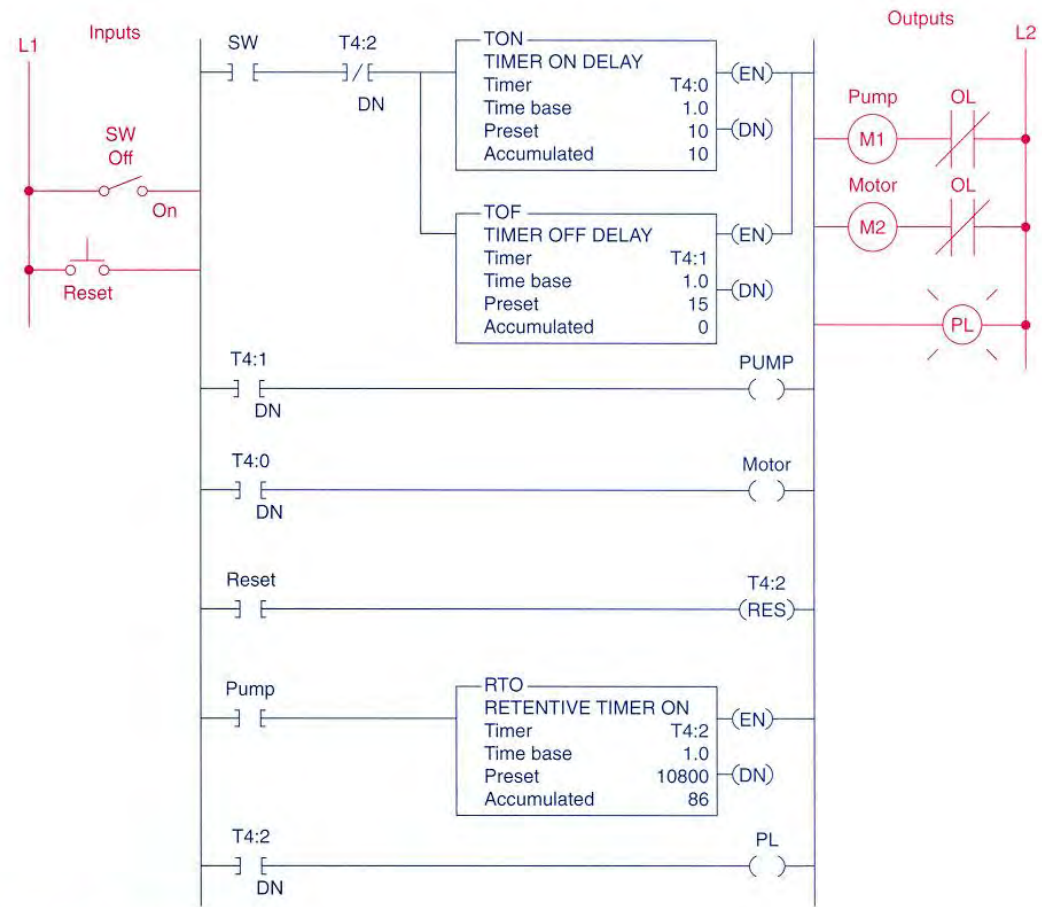
Retentive Timers

Animated demonstration:
(search on the Schneider PLC or discuss implementation)

Ladder diagram

Example:

• SW ON to start operation

• Before motor starts, lubrificate 10 s with oil.

• SW OFF to stop. (lubrificate 15 s more).

• After 3 hours of pump operation, stop motor and signal with pilot light.
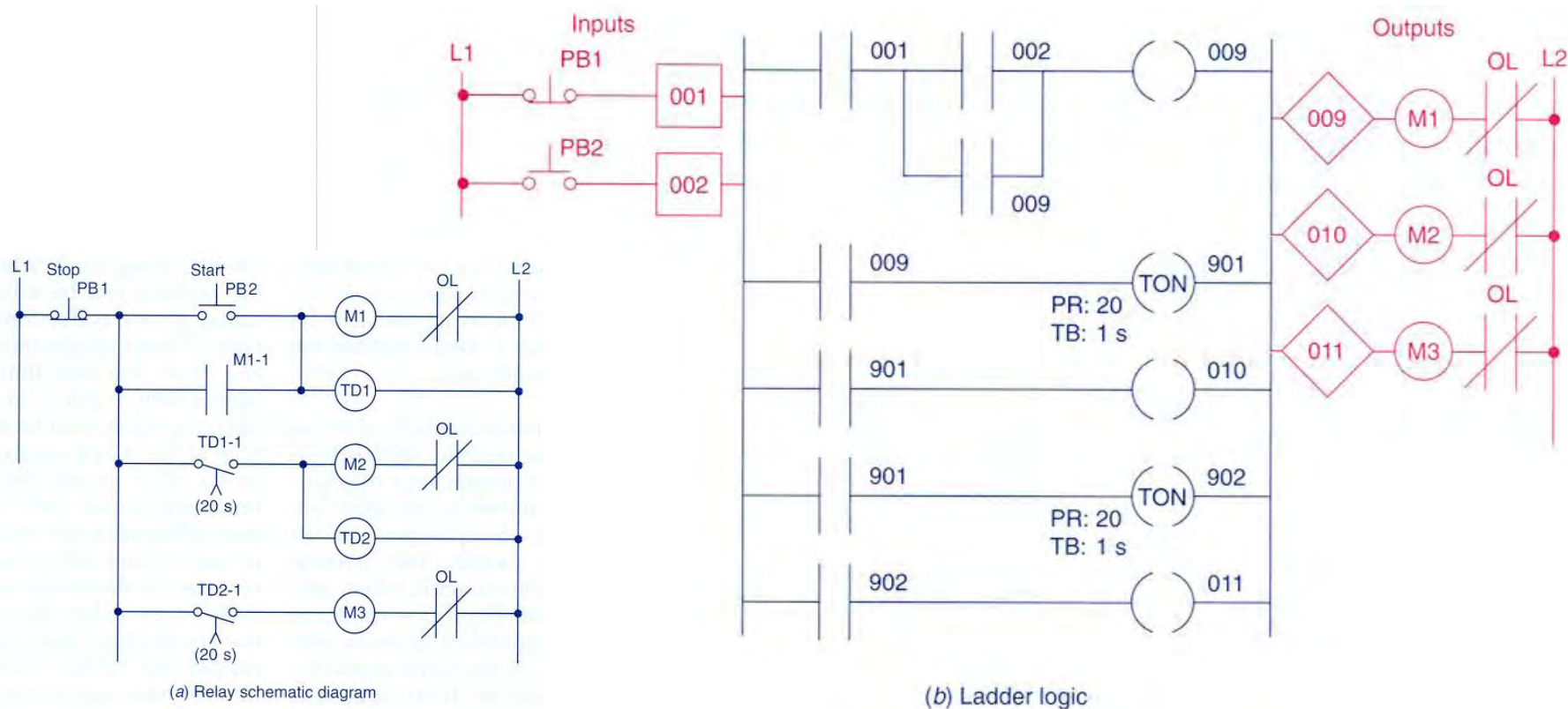
• Reset available after servicing.

Fig. 7-23
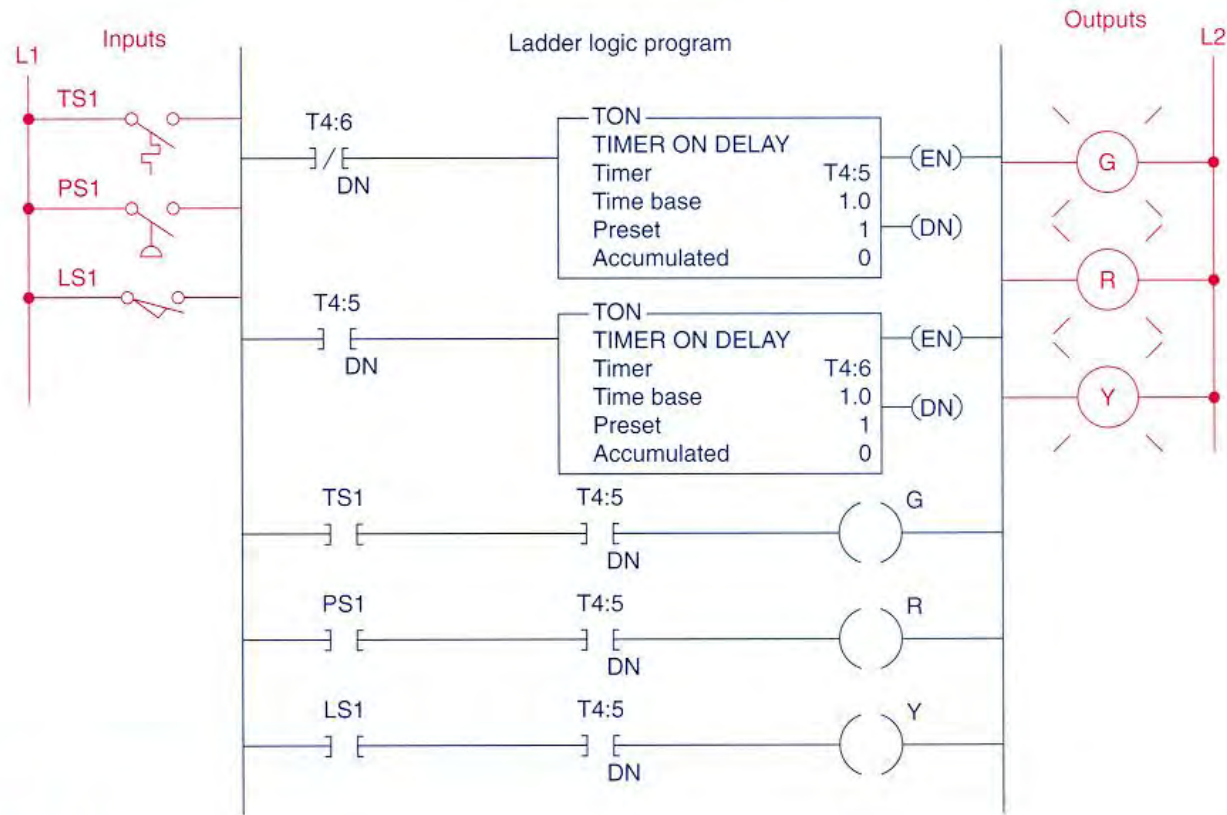Bearing lubrication program.

Ladder diagram

Cascaded Timers

(a) Relay schematic diagram

(b) Ladder logic

Fig. 7-24

Sequential time-delayed motor-starting circuit.

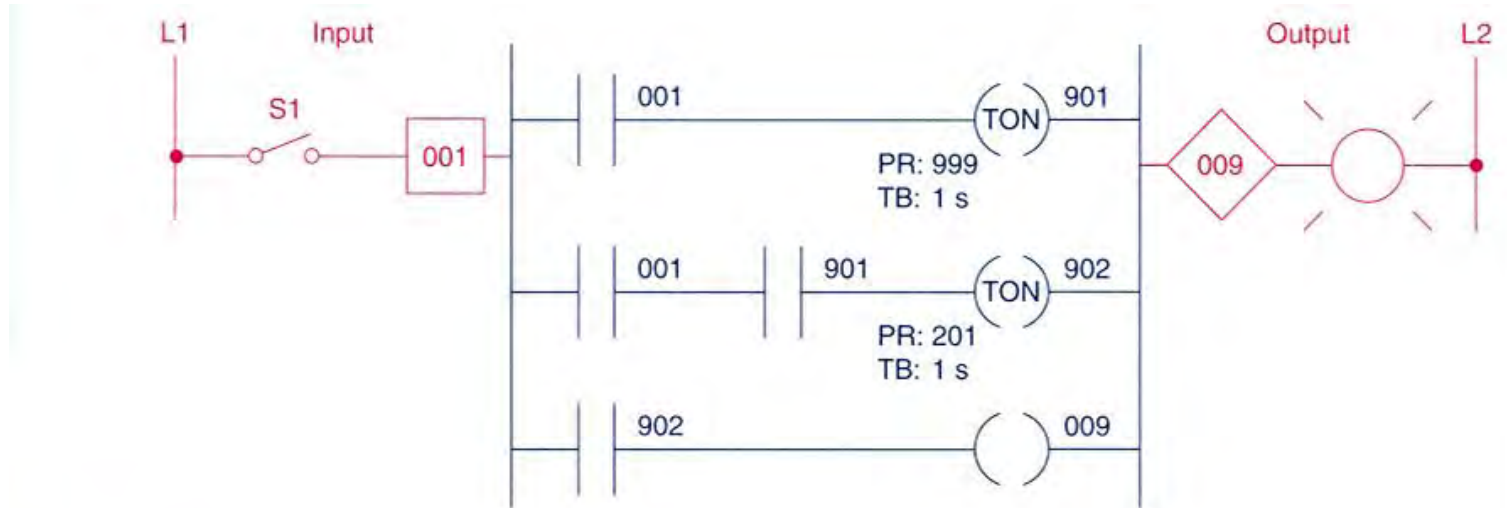# Ladder diagram

Cascaded Timers (bistable system)

Fig. 7-25

Annunciator flasher program.

Ladder diagram

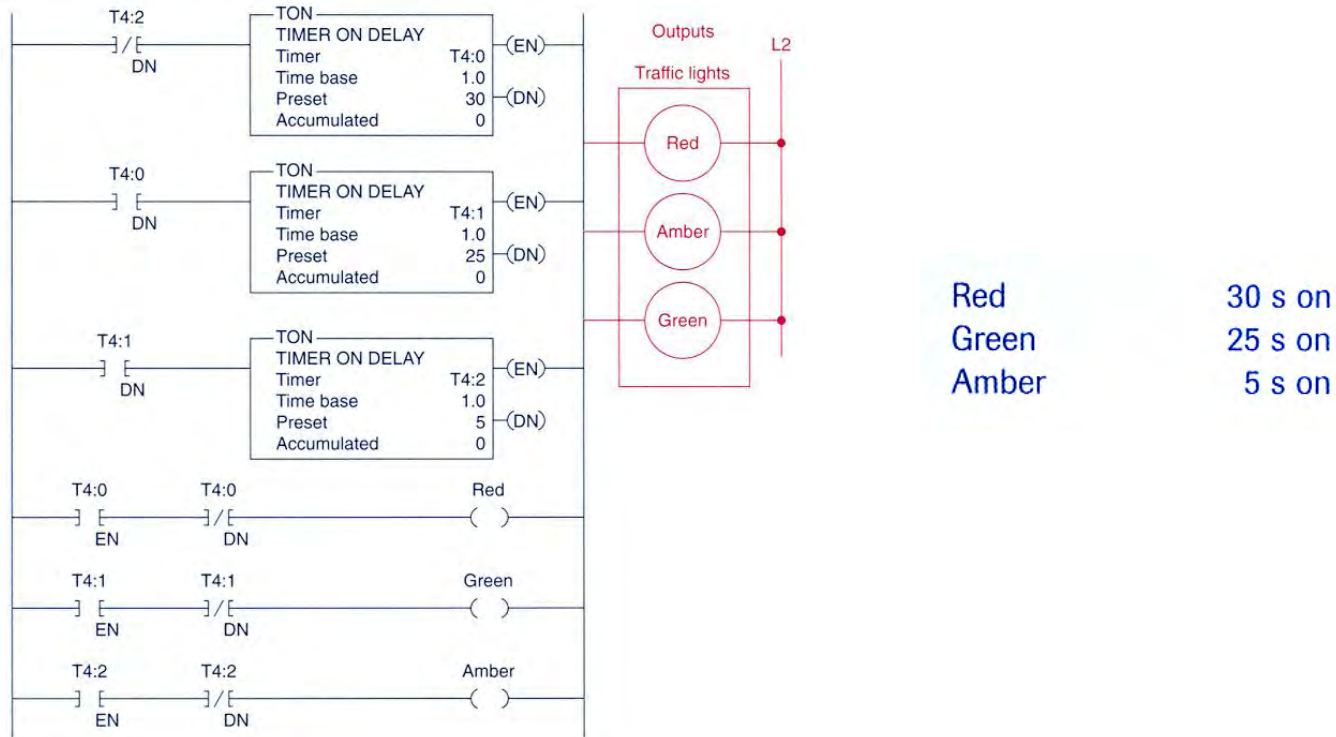## Timers for very long time intervals

Fig. 7-26
Cascading of timers for longer time delays.
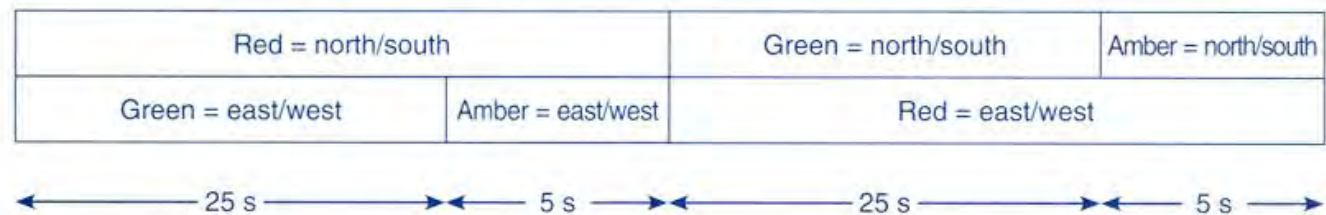
Ladder diagram

Example of a semaphore

Fig. 7-27
Control of traffic lights in one direction.

Example of a semaphore in both directions

Red	30 s on
Green	25 s on
Amber	5 s on

Red = north/south		Green = north/south	Amber = north/south
Green = east/west	Amber = east/west	Red = east/west	

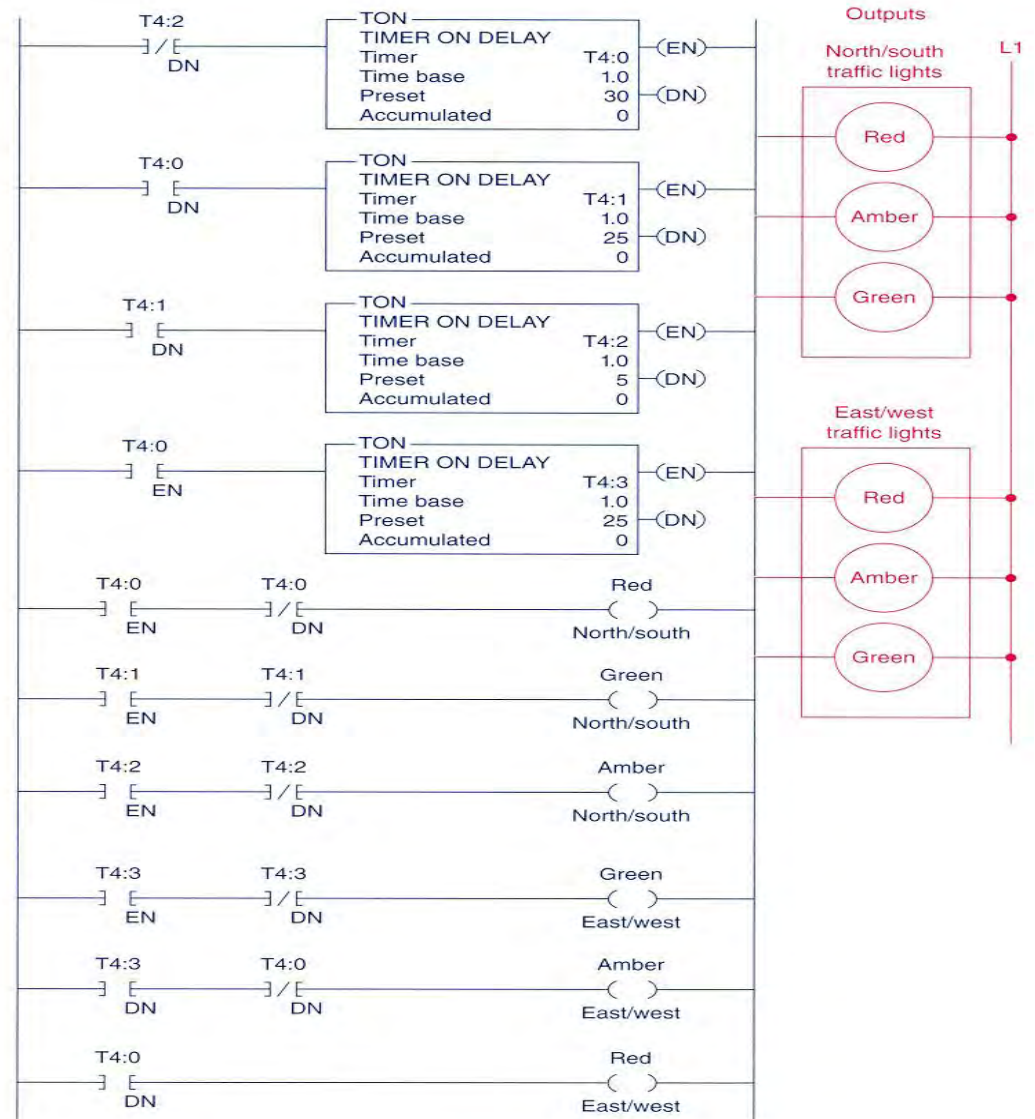←———— 25 s ————→ ←— 5 s —→ ←———— 25 s ————→ ←— 5 s —→

(b) Timing chart

Fig. 7-28 (continued)

Control of traffic lights in two directions.

Example of a semaphore

in both directions

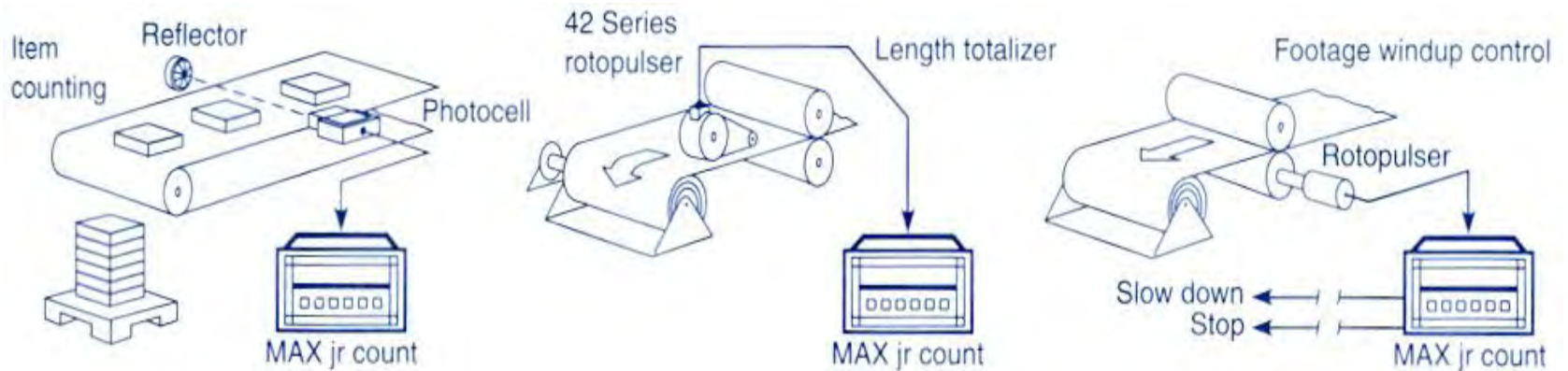(a) Ladder logic

Fig. 7-28
Control of traffic lights in two directions.
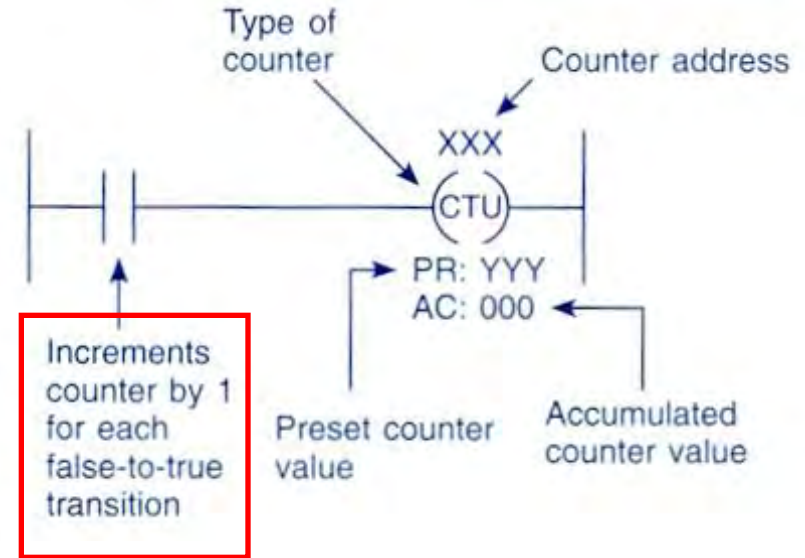
Ladder diagram

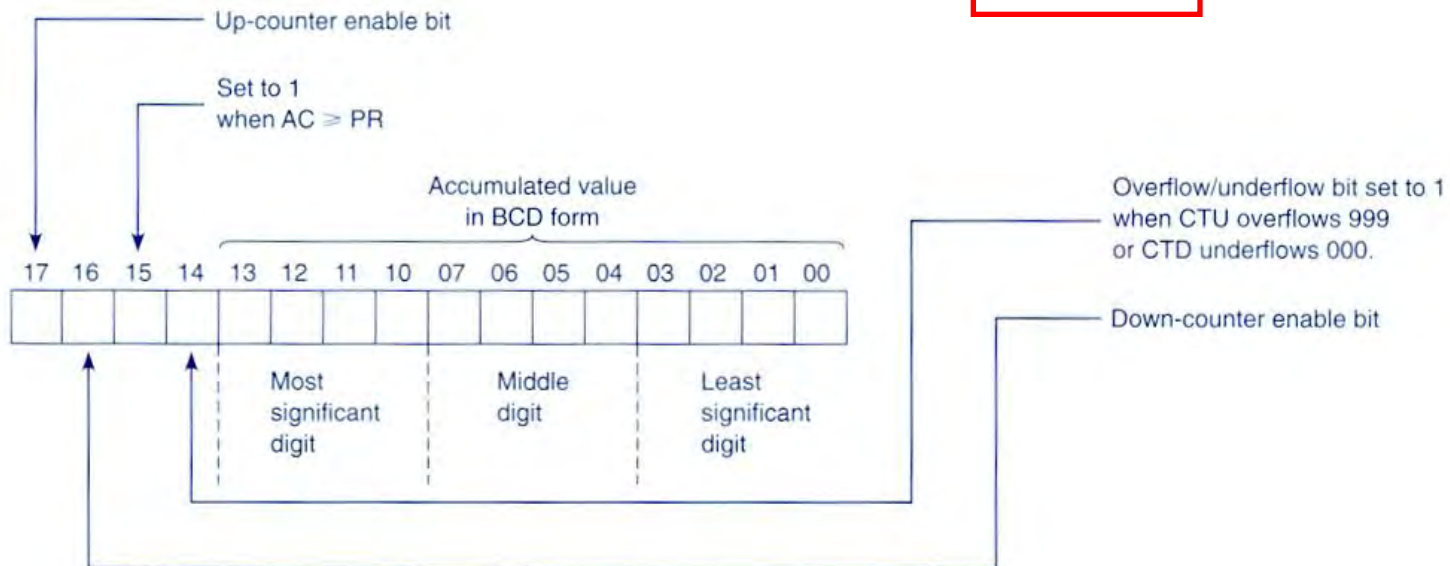## Counters

Some applications...

Fig. 8-3

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

Ladder diagram

Implementation of Counters
in the PLC-5 of *Allen-Bradley*:
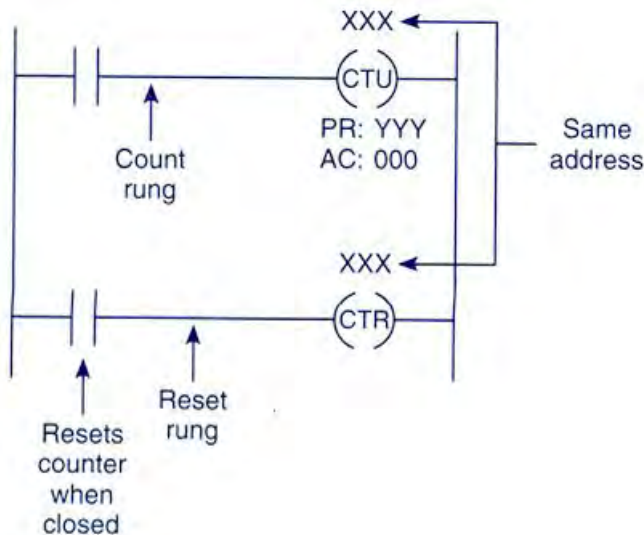
Internal structure representation

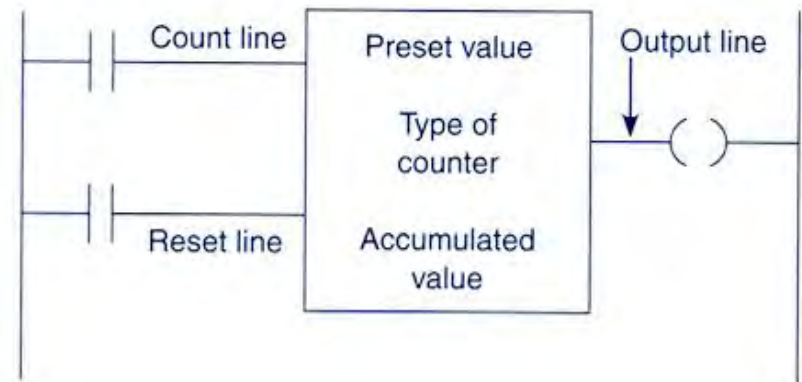(b) Allen-Bradley PLC-2 timer accumulated value word
(bit addressing is in octal)

Ladder diagram

Implementation of Counters
in the PLC-5 of *Allen-Bradley*:
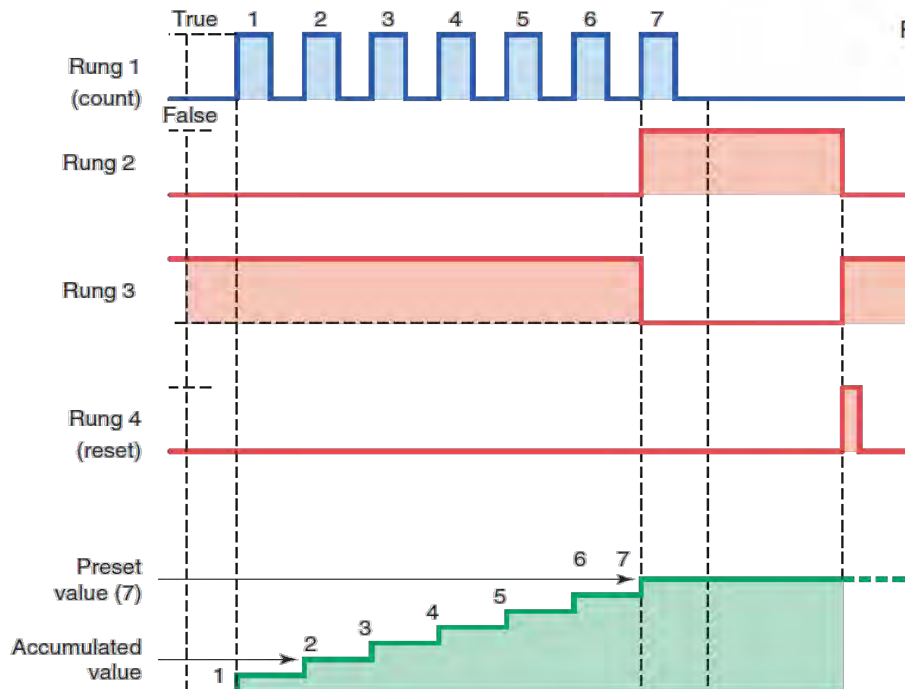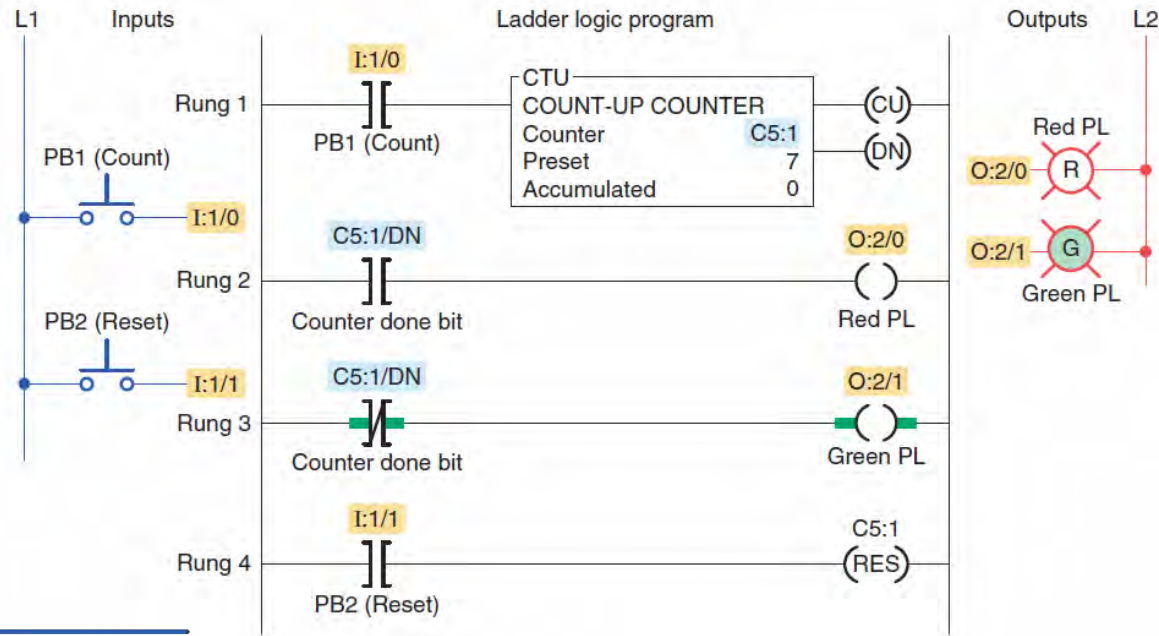
Two alternative representations:

Coil-formatted counter and reset
instructions

Block-formatted counter
instruction

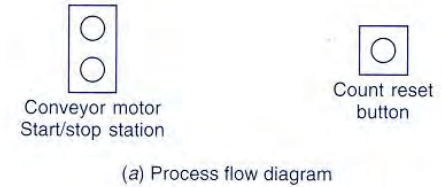# Ladder diagram

Up-counters

Usage of an **incremental up-counter** and the corresponding **temporal diagram:**

PB1 increments counting
PB2 resets the counting

Ladder diagram

Example

(a) Process flow diagram

(b) Program

Fig. 8-13
Conveyor motor program.

1. Start conveyor motor
2. Passing cases increment counter
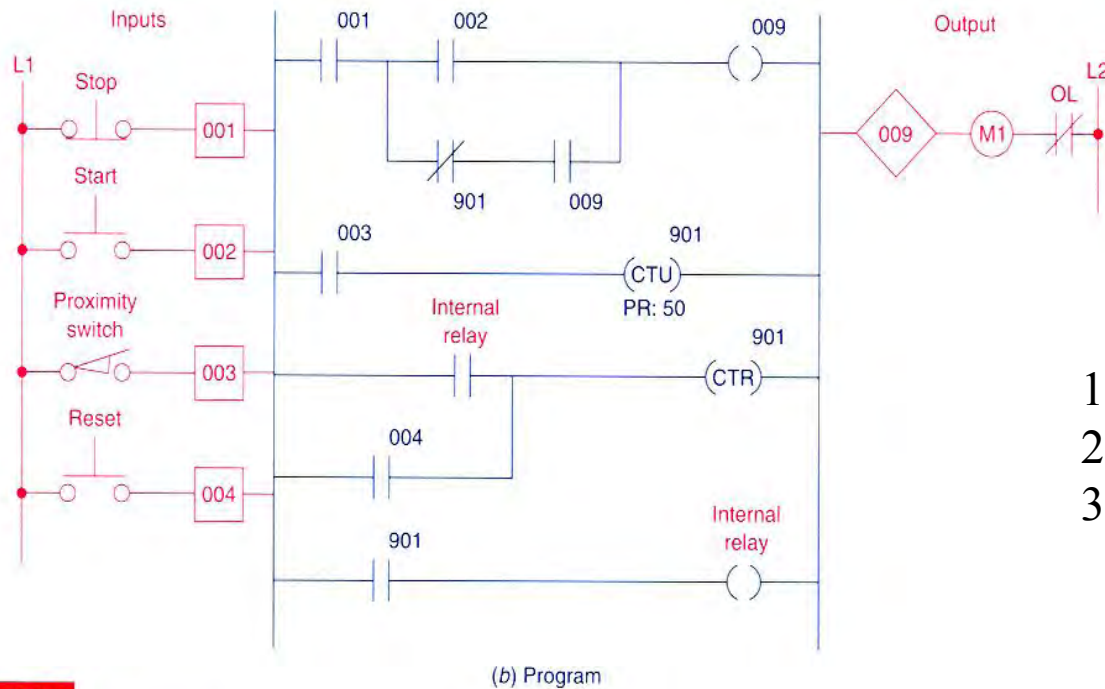3. After 50 cases, stop motor

Ladder diagram
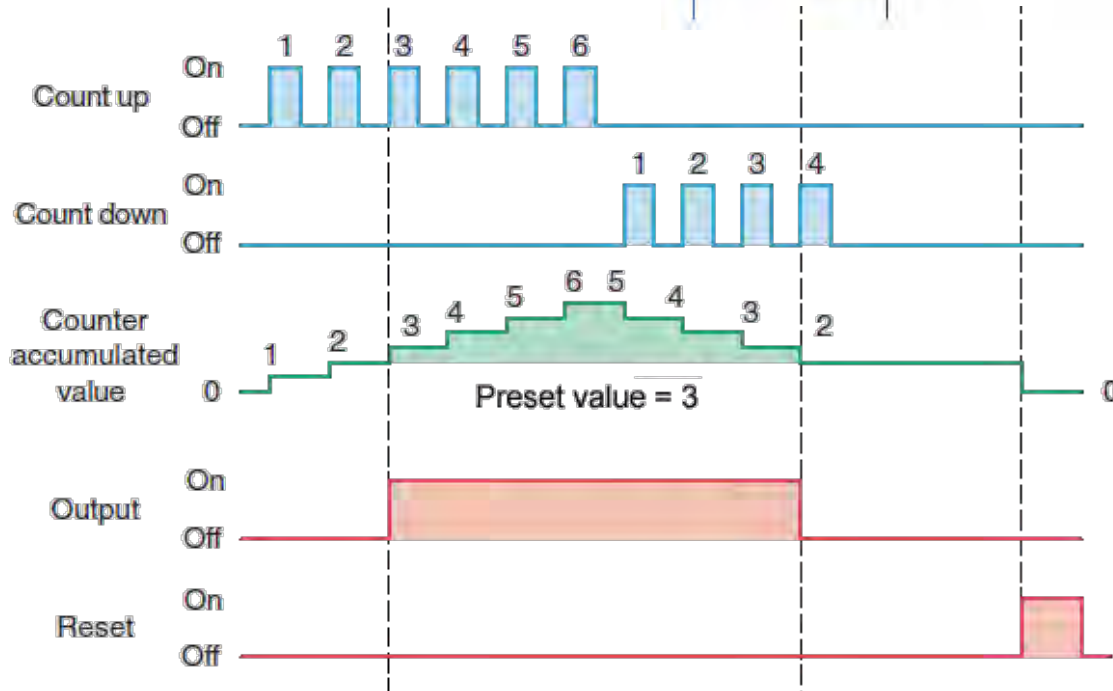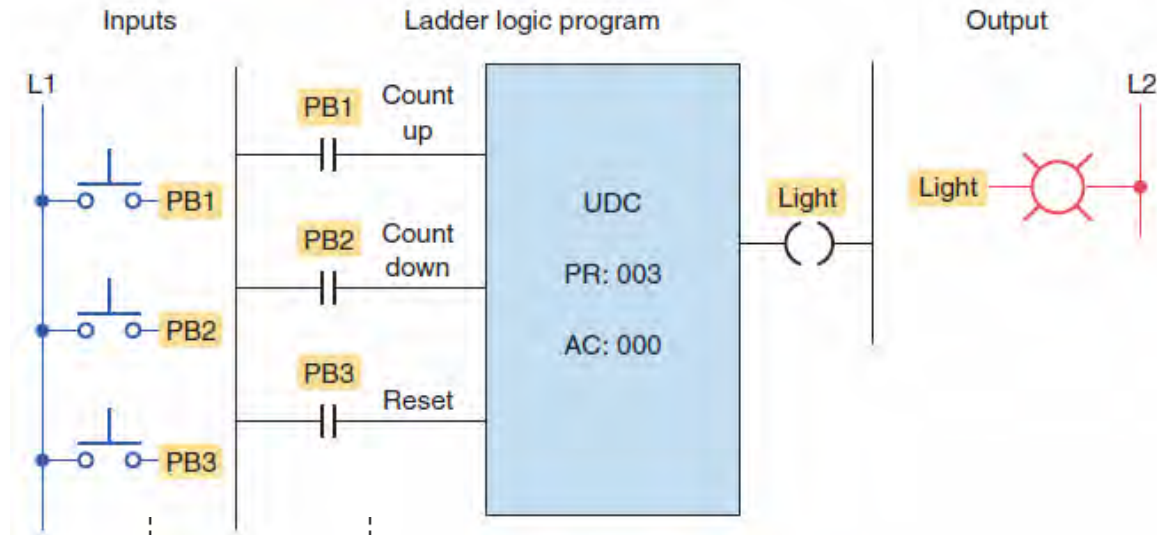
Up/down-counters
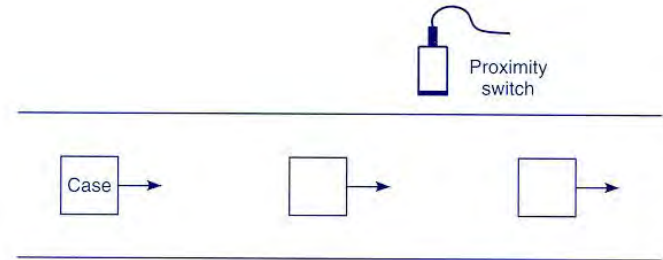

Usage of an **incremental up-down-counter** and the corresponding **temporal diagram:**

PB1 increments counting
PB2 decrements the counting
PB3 resets the counter

Ladder diagram

Up/down-counters

Example:

Finite parking garage

Fig. 8-17

Parking garage counter.
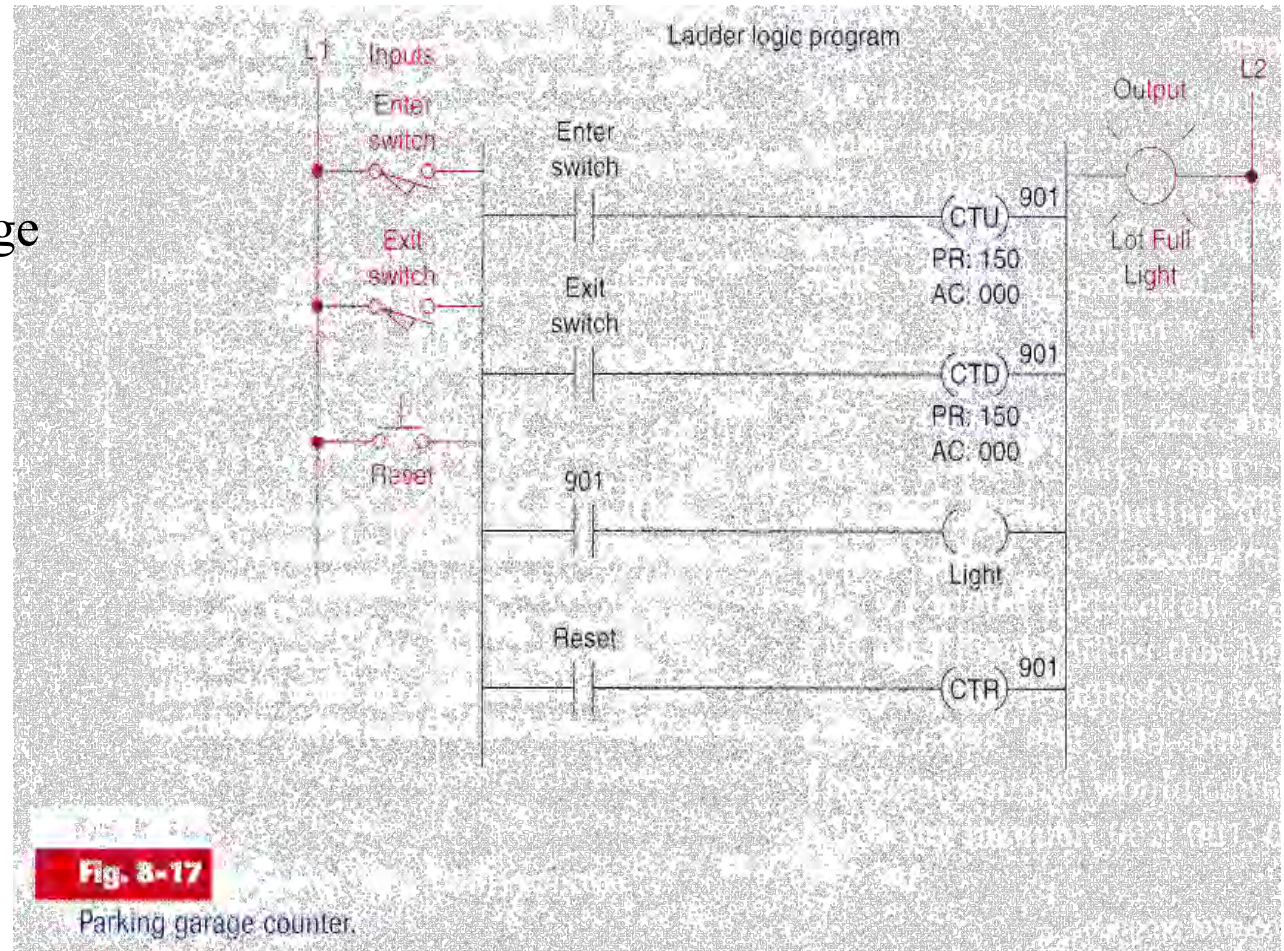
Ladder diagram

Cascaded Counters
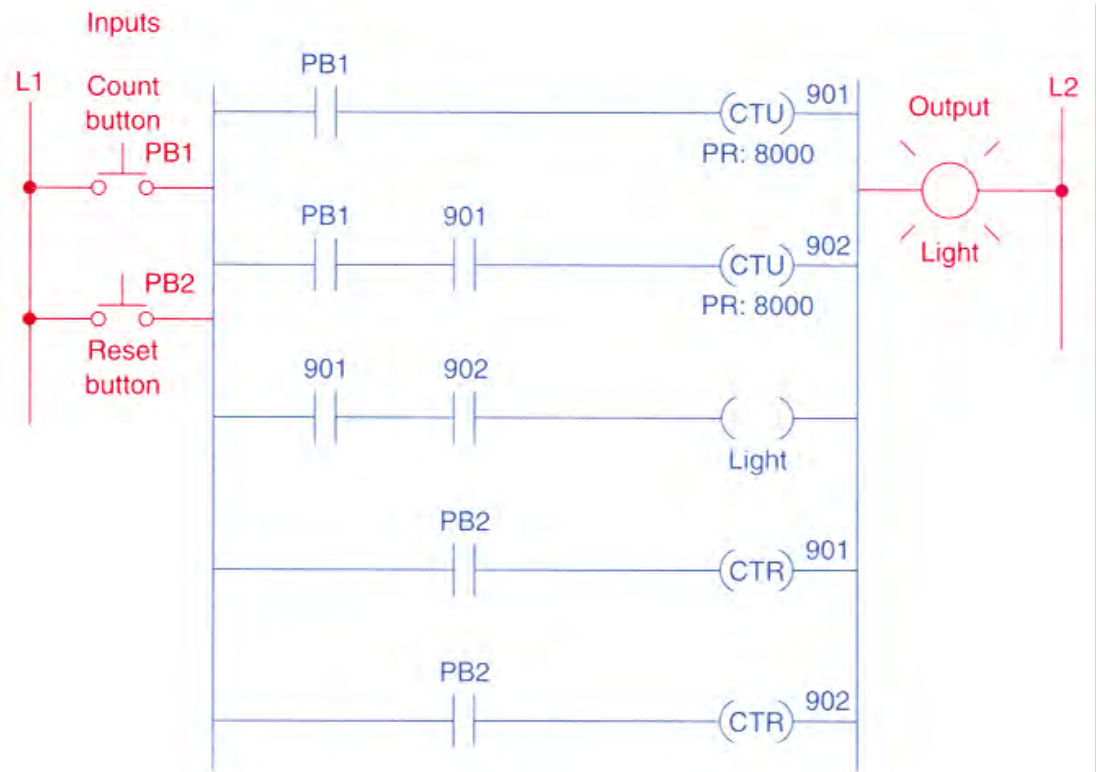
Example:

Fig. 8-21

Counting beyond the maximum count.

Ladder diagram

Cascaded Counters

Example:

24 hours clock
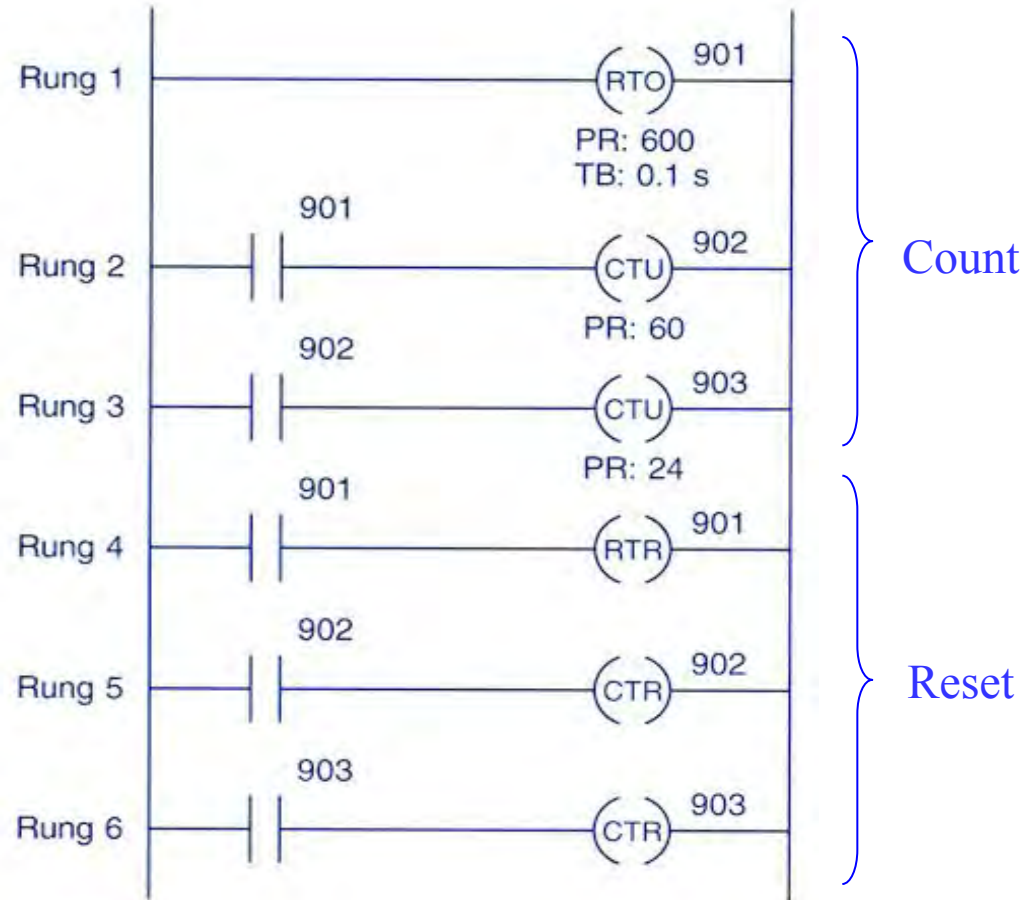
Fig. 8-23

A 24-h clock program.

Ladder diagram

Incremental *Encoder*

counter measures **rotation angle** or **rotation speed** (if divided by time)
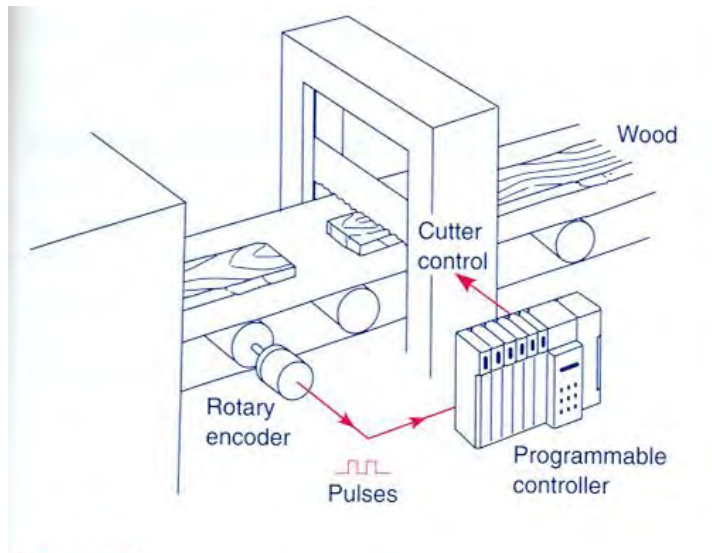
Fig. 8-26
Cutting objects to a specified size.

Incremental disk

Electronics board

Photodetector assembly

Mask

Code disk

Light source

Bearing housing assembly

Output waveform

Code track on disk
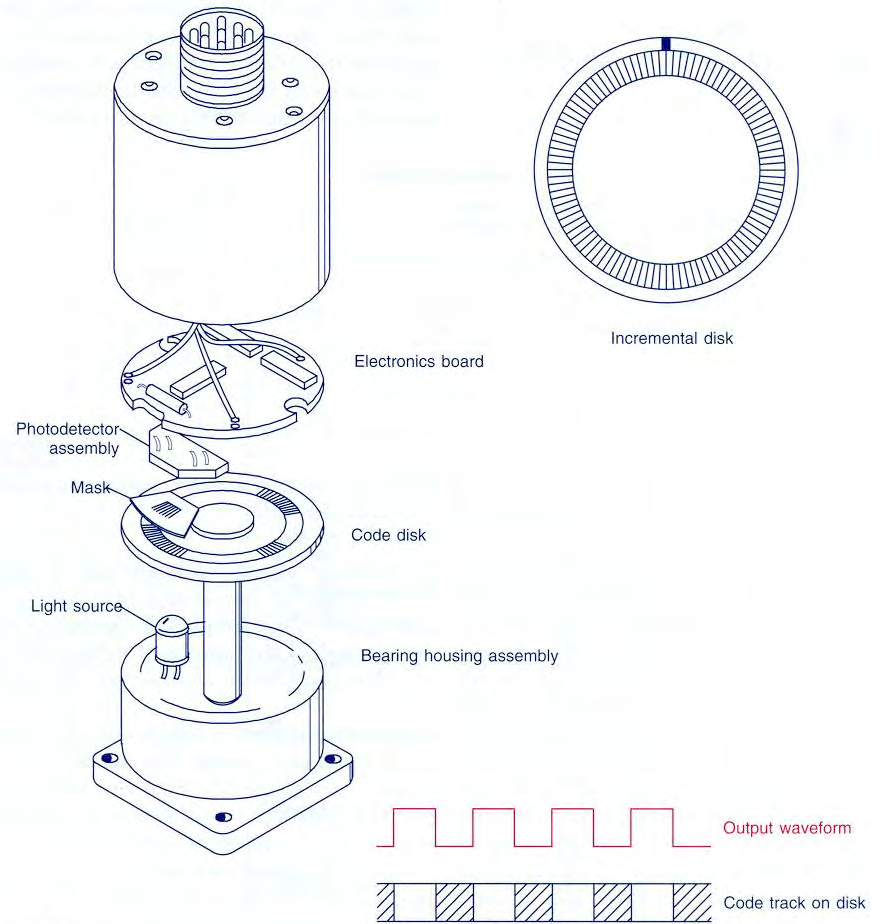
Fig. 8-25
Incremental encoder. *(Courtesy of BEI Motion Systems Company.)*

Ladder diagram

Incremental *Encoder*

Example:
counter as a "length sensor"
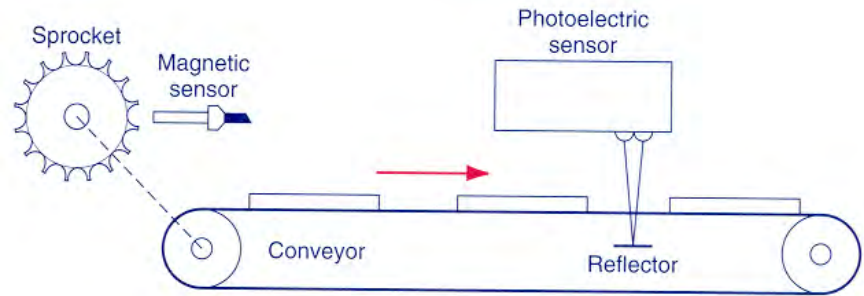
(a) Process
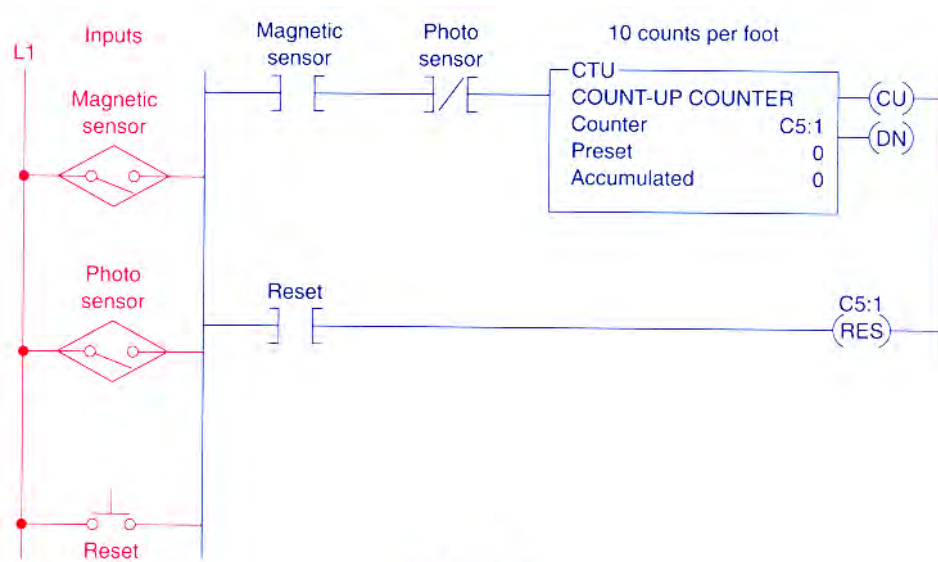
Fig. 8-27
Counter used for length measurement.

Ladder diagram

Counters in PL7

Characteristics:
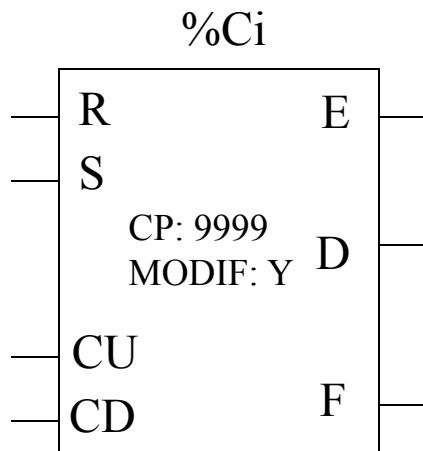
Identifier: %Ci   0..31 in the TSX37

%Ci

R	E
S	
CP: 9999 MODIF: Y	D
CU	
CD	F

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

Ladder diagram

Counters in Unity Pro

CTU_Instance
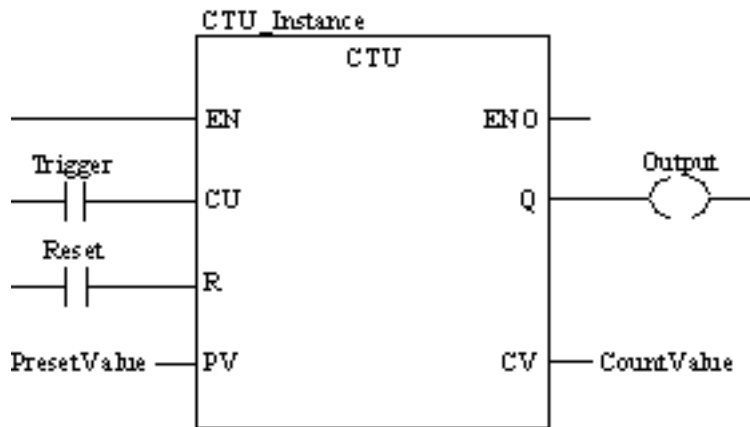
CU "0" to "1" => CV is incremented by 1

CV ≥ PV => Q:=1

R=1 => CV:=0

CTUD_Instance
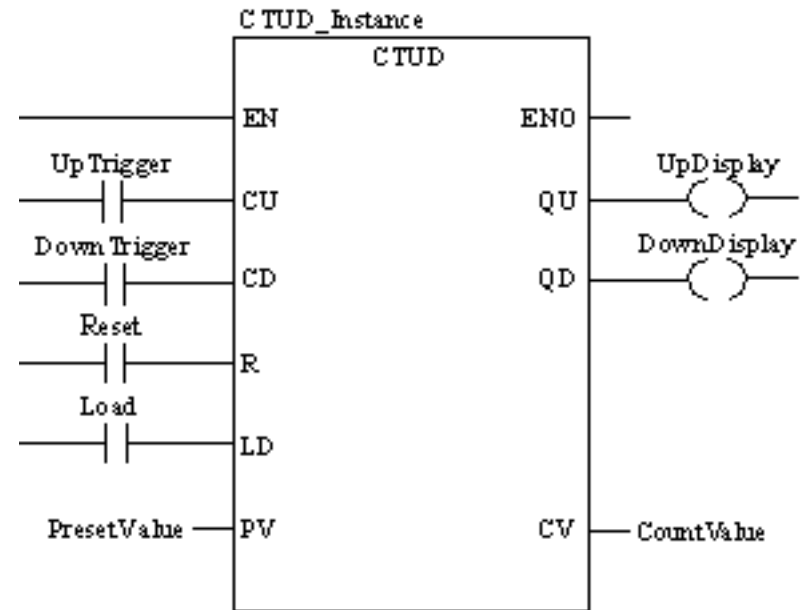
CU "0" to "1" => CV is incremented by 1
CD "0" to "1" => CV is decremented by 1

CV ≥ PV => QU:=1
CV ≤ 0 => QD:=1

R=1 => CV:=0   **LD=1** => CV:=PV
R has precedence over LD

NOTE: counters are saturated such that no overflow occurs

Page 77

Ladder diagram

Numerical Processing

Algebraic and Logic Functions

Ladder diagram

Numerical Processing

Arithmetic Functions

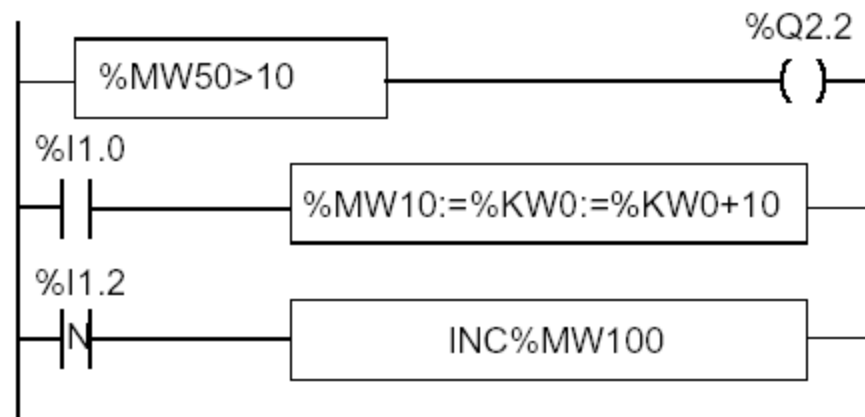| +	addition of two operands	**SQRT**	square root of an operand
-	subtraction of two operands	**INC**	incrementation of an operand
*	multiplication of two operands	**DEC**	decrementation of an operand
/	division of two operands	**ABS**	absolute value of an operand
REM	remainder from the division of 2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

Ladder diagram

Numerical Processing

Example:

Arithmetic functions
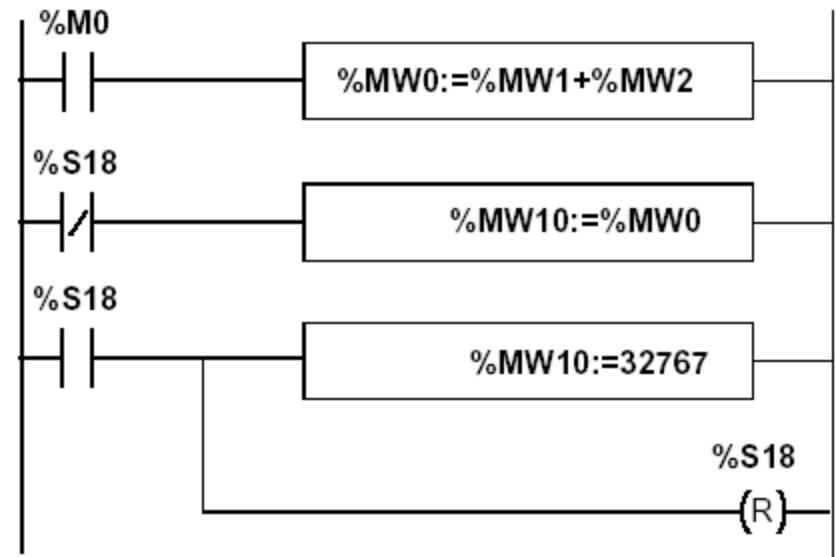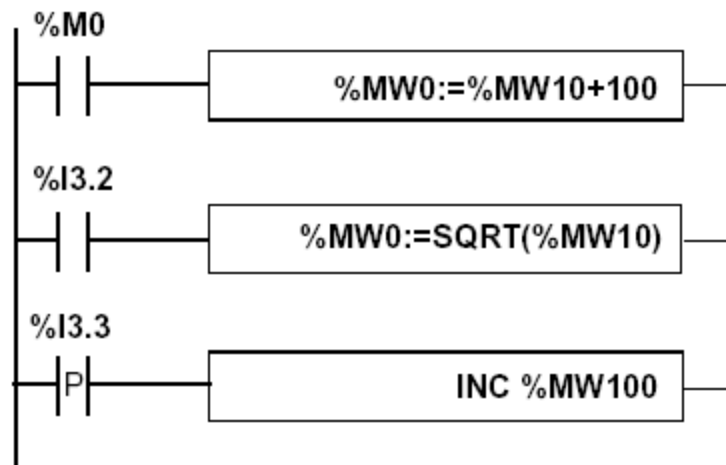
Use of a system variable:

%S18 – flag de overflow

Ladder diagram

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.

Ladder diagram

Numerical Processing

Example:

Logic functions
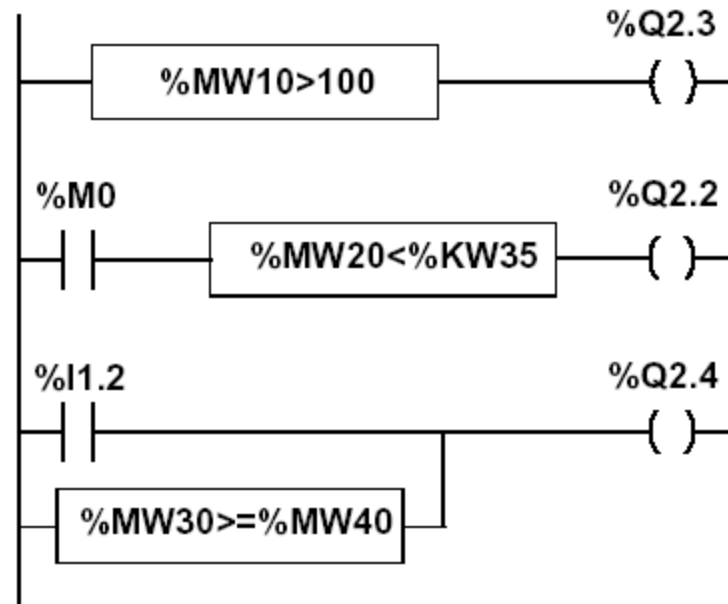
Ladder diagram

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

Ladder diagram

Structures for Control of Flux

Subroutines

Call and Return
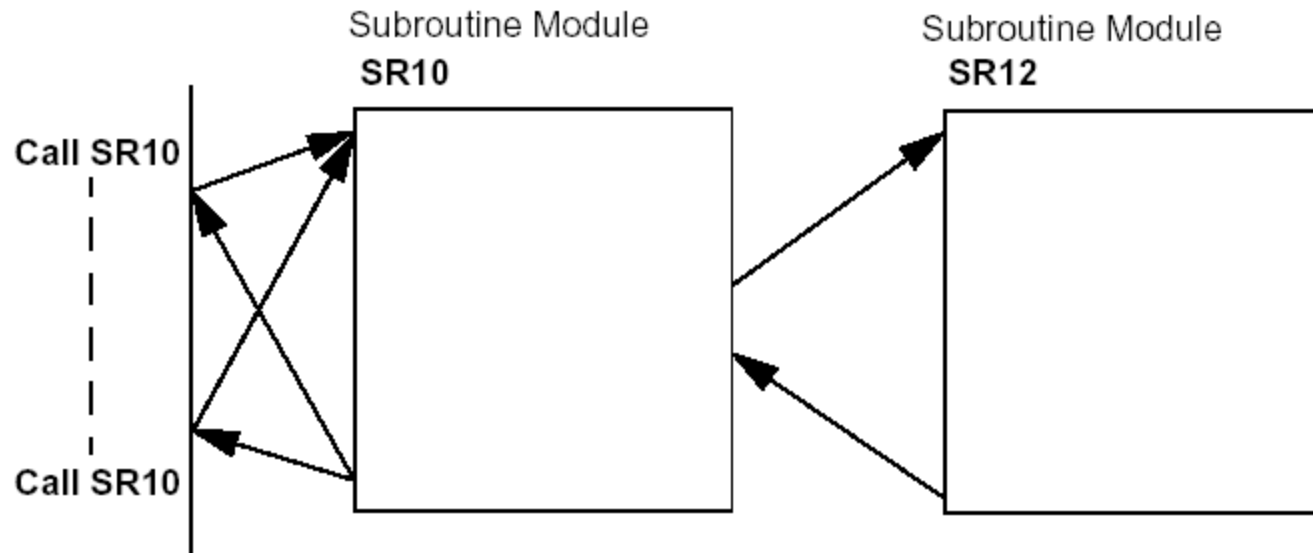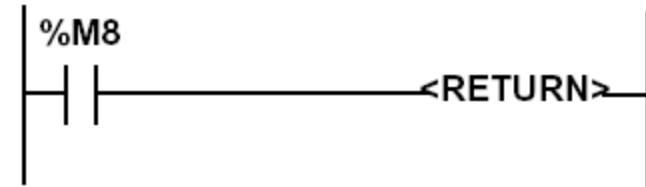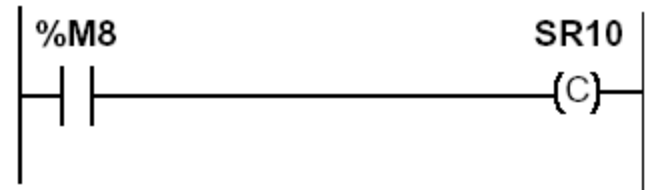
Ladder diagram

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)
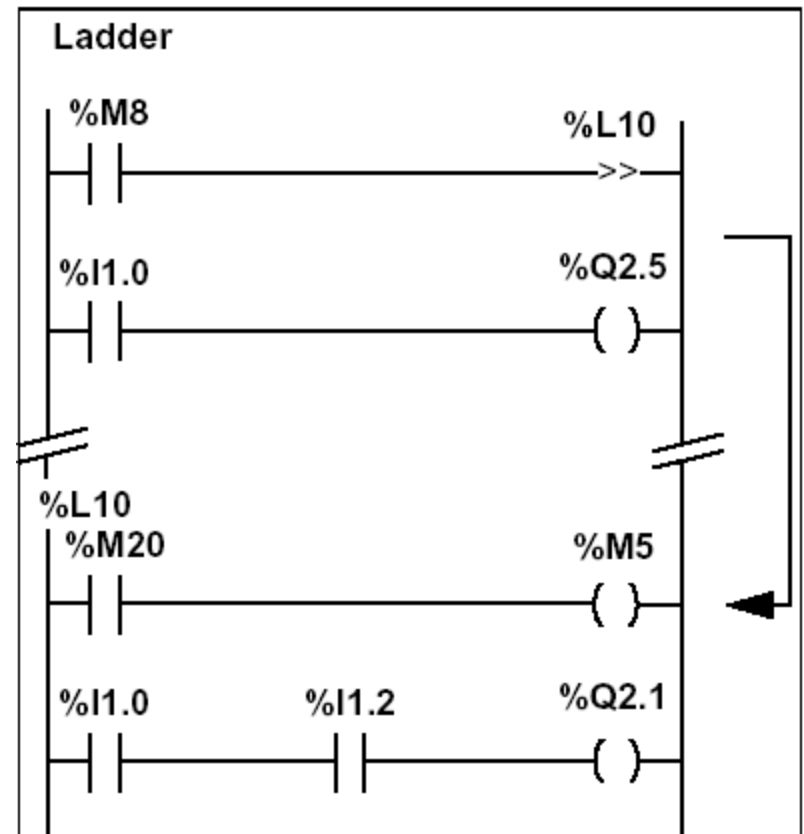

Ladder diagram

Structures for Control of Flux

Example:

Use of jump instructions

Attention to:

• **INFINITE LOOPS ...**

• **It is not a good style of programming!...**

• **Does not improove the legibility of the proposed solution.**

Ladder diagram

Structures for Control of Flux

Halt
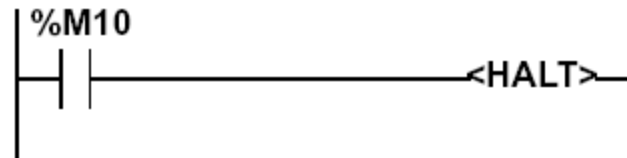
Stops all processes!

Events masking

Ladder diagram

There are other advanced instructions (see manual)

- **Monostable**

- **Registers of 256 words (LIFO ou FIFO)**

- *DRUMs*

- **Comparators**

- *Shift-registers*

...

- **Functions to manipulate** *floats*

- **Functions to convert bases and types**

Ladder diagram
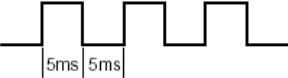
Numerical Tables

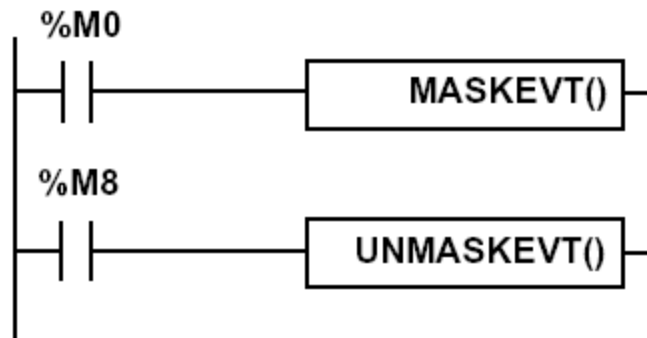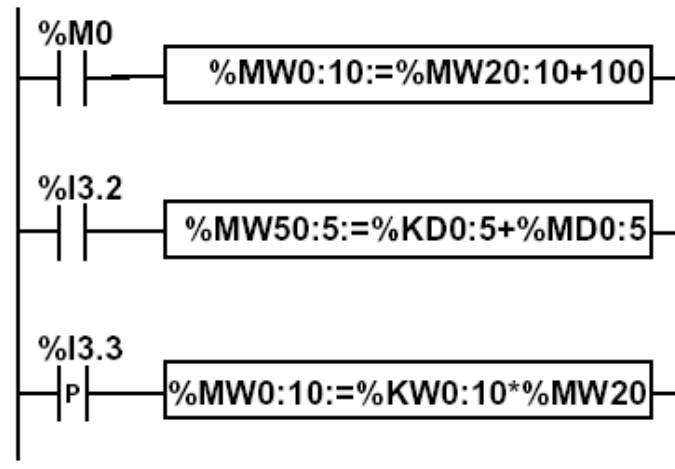| Type	Format	Maximum address	Size	Write access
Internal words	Simple length	%MWi:L	i+L<=Nmax (1)	Yes
	Double length	%MWDi:L	i+L<=Nmax-1 (1)	Yes
	Floating point	%MFi:L	i+L<=Nmax-1 (1)	Yes
Constant words	Single length	%KWi:L	i+L<=Nmax (1)	No
	Double length	%KWDi:L	i+L<=Nmax-1 (1)	No
	Floating point	%KFi:L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

```
%M0
 | |        %MW0:10:=%MW20:10+100


%I3.2
 | |        %MW50:5:=%KD0:5+%MD0:5


%I3.3
 |P|        %MW0:10:=%KW0:10*%MW20
```

Ladder diagram

System information: system bits

Bit	Function	Description	Initial state	TSX37	TSX57
%S0	Cold start	Normally on 0, this bit is set on 1 by: ● loss of data on power restart (battery fault), ● the user program, ● the terminal, ● cartridge uploading, ● pressing on the RESET button. This bit goes to 1 during the first complete cycle. It is reset to 0 before the following cycle. (Operation)	0	YES	YES
%S1	Warm restart	Normally on 0, this bit is set on 1 by: ● power restart with data save, ● the user program, ● the terminal. It is reset to 0 by the system at the end of the first complete cycle and before output is updated. (Operation)	0	YES	YES
%S4	Time base 10ms	An internal timer regulates the change in status of this bit. It is asynchronous in relation to the PLC cycle. Graph : 5ms 5ms	-	YES	YES
%S5	Time base 100 ms	Idem %S4	-	YES	YES
%S6	Time base 1 s	Idem %S4	-	YES	YES
%S7	Time base 1 mn	Idem %S4	-	YES	YES

See manual for the remaining 100 bits generated...

Ladder diagram

System information: system *words*

Words	Function	Description	Management
%SW0	Master task scanning period	The user program or the terminal modify the duration of the master task defined in configuration. The duration is expressed in ms (1.255 ms) %SW0=0 in cyclic operation. On a cold restart: it takes on the value defined by the configuration.	User
%SW1	Fast task scanning period	The user program or the terminal modify the duration of the fast task as defined in configuration. The duration is expressed in ms (1.255 ms) On a cold restart: it takes on the value defined by the configuration.	User
%SW8	Acquisition of task input monitoring	Normally on 0, this bit can be set on 1 or 0 by the program or the terminal. It inhibits the input acquisition phase of each task. ● %SW8:X0 =1 assigned to MAST task: outputs linked to this task are no longer guided. ● %SW8:X1 =1 assigned to FAST task: outputs linked to this task are no longer guided.	User
%SW9	Monitoring of task output update	Normally on 0, this bit can be set on 1 or 0 by the program or the terminal. Inhibits the output updating phase of each task. ● %SW9:X0 =1 assigned to MAST task: outputs linked to this task are no longer guided. ● %SW9:X1 =1 assigned to FAST task: outputs linked to this task are no longer guided.	User
%SW10	First cycle after cold start	If the bit for the current task is on 0, this indicates that the first cycle is being carried out after a cold start. ● %SW10:X0: is assigned to the MAST Master task ● %SW10:X1: is assigned to the FAST fast task	System
%SW11	Watchdog duration	Reads the duration of the watchdog as set in configuration. It is expressed in ms (10...500 ms).	System

**See manual
for the remaining
140 words generated...**

Page 91

Ladder diagram   Software Organization

```
Project Browser                    [X]
Structural View
⊟ 📁 Program
  ⊟ 📁 Tasks
    ⊟ 📁 MAST
      ⊟ 📁 Section
          📄 Counting
          📄 Drilling
        ⊞ SFC Washing
      ⊟ 📁 SR Sections
          ST SR1
          LD SR2
    ⊟ 📁 FAST
        📁 Sections
        📁 SR Sections
    ⊟ 📁 AUX0
        📁 Sections
        📁 SR Sections
  ⊟ 📁 Events
    ⊟ 📁 Timer event
        ST Timer0
        ST Timer1
    ⊟ 📁 I/O Events
        ST Evt0
        ST Evt1
◄                                  ►
```

A program can be built from:
   Tasks, that are executed cyclically or periodically.

Tasks **MAST** / FAST / AUX are built from:
   Sections
   Subroutines
   Event processing, that is carried out before all other tasks.

Event processing is built from:
   Sections for processing time controlled events
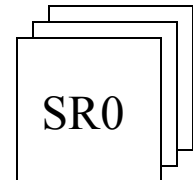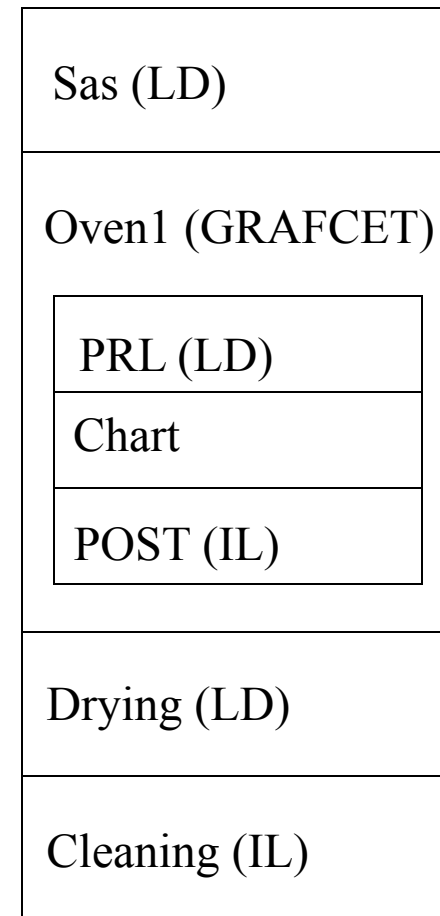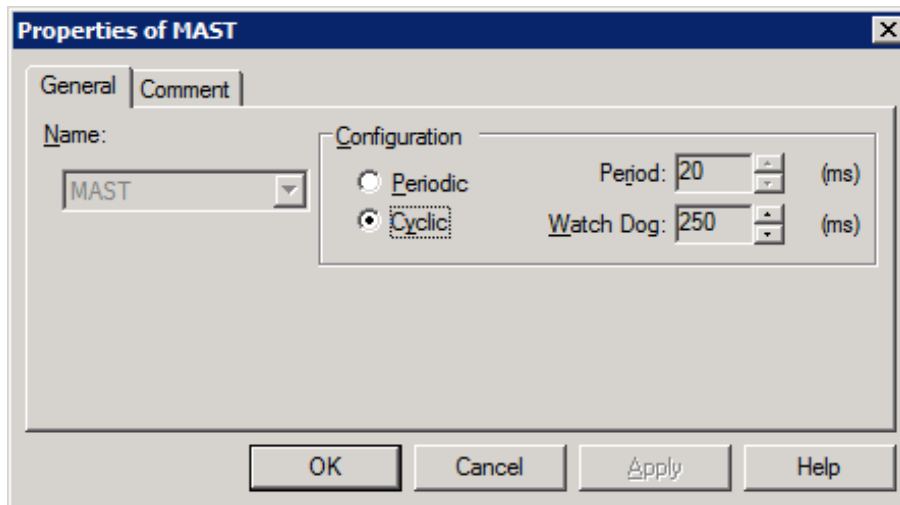   Sections for processing hardware controlled events

Unity - Project Browser

Page 92

Ladder diagram　　**Software Organization**

MAST – Master Task Program

　　Composed by sections

　　Execution Cyclic or Periodic

Properties of MAST
General | Comment
Name:	Configuration
MAST	○ Periodic　Period: 20 (ms)
	⊙ Cyclic　Watch Dog: 250 (ms)
	OK　Cancel　Apply　Help

Sas (LD)
Oven1 (GRAFCET)
PRL (LD)
Chart
POST (IL)
Drying (LD)
Cleaning (IL)

SR0

Ladder diagram   Software Organization

FAST – Fast Task Program

Priority greater than MAST
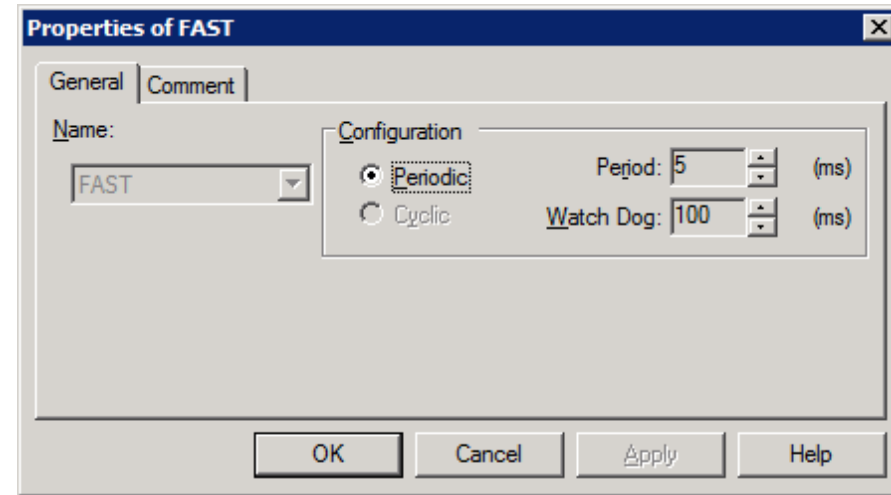
• Executed Periodically (1-255ms)

• Verified by a *Watchdog,* impacts on %S11

• %S31 *Enables* or *disables* a FAST

• %S33 gives the execution time for FAST

Ladder diagram　　Software Organization

Event Processes **– Processes that can react to external changes**
(16 in the Micro 3722 EV0 to EV15)
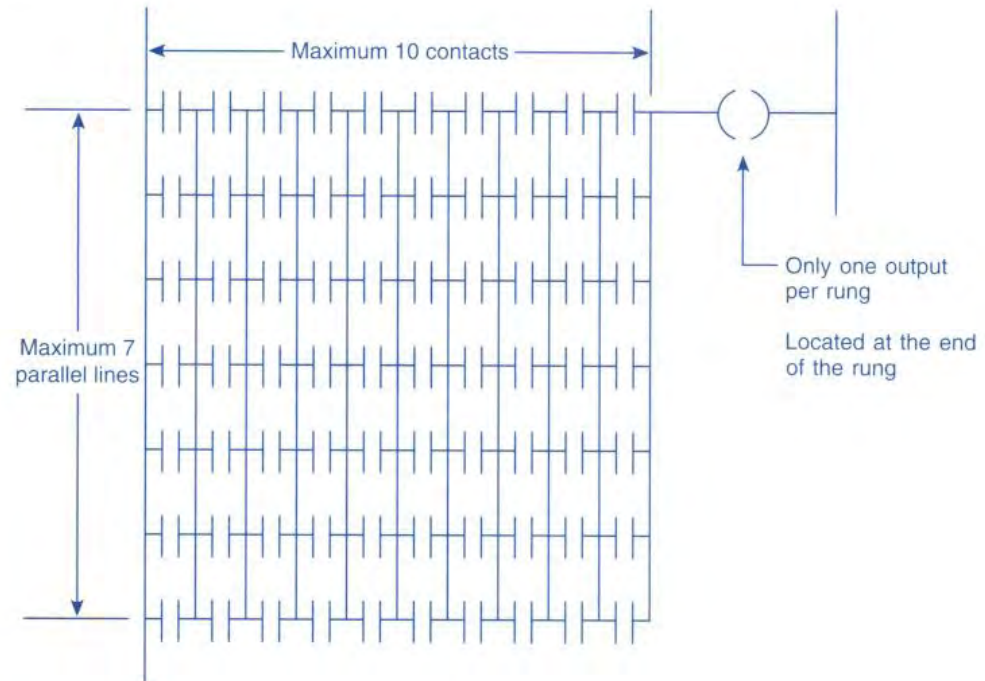
Priority greater than MAST and FAST!

Event Generators

• **Inputs 0 to 3 in module 1, given transitions**

• **Counters**

• **Upon telegrams reception**

• **%S38 *Enables* or *disables* event processes**

(also with MASKEVT() or UNMASKEVT())

Ladder diagram   Development tools

Each PLC has limitations in terms of connections

Example:

Fig. 5-27

Typical PLC matrix limitation diagram. The exact limitations are dependent on the particular type of PLC used. Programming more than the allowable series elements, parallel branches, or outputs will result in an error message being displayed.

Ladder diagram   Development tools

It is important to learn the potentialities and ...

the limitations of the developing tools,

i.e. *STUDYING the manuals is a MUST.*

Ladder diagram　　　**Development tools**

Last but not least, *learn how to develop and debug programs*
(and how to do some fine tuning).

Ladder diagram   Development tools

Last but not least, *learn how to develop and debug programs*
(and how to do some fine tuning)**.**