Industrial Automation

(Automação de Processos Industriais)

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

Faculty:

Prof. Paulo Jorge Oliveira pjcro @ isr.ist.utl.pt Tel: 21 8418053

Prof. José Gaspar jag @ isr.ist.utl.pt Tel: 21 8418293

Objectives:

- Analysis of systems for industrial automation.

- Methodologies for the implementation of solutions in industrial automation.

- Programming languages of PLCs (Programmable Logic Controllers).

- CAD/CAM and Computerized Numerical Controlled machines.

- Discrete Event Systems Modeling.

- Supervision of Processes in Industrial Automation.

Syllabus:

Chap. 1 – Introduction to Automation [1 week]

Introduction to components in industrial automation.
Introduction to methodologies for problem modeling.
Cabled logic versus programmed logic.

Chap. 2 – Introduction to PLCs [2 weeks]

Components of Programmable Logic Controllers (PLCs).
Internal architecture and functional structure.
Input / output Interfaces. Interconnection of PLCs .

Chap. 3 – PLCs Programming Languages [2 weeks]

Standard languages (IEC-1131-3):
Ladder Diagram; Instruction List and *Structured Text*.
Software development resources.

Syllabus (cont.):

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

The GRAFCET norm. Elements of the language.
Modeling techniques using GRAFCET.

Chap. 5 – CAD/CAM and CNC Machines [1 week]

Methodology CAD/CAM. Types of Computerized Numerical Controlled machines. Interpolation of trajectories.
Flexible fabrication cells.

Chap. 6 – Discrete Event Systems [1 week]

Modeling of discrete event systems (DESs).
Automata. Petri networks. State and dynamics of PNs.

Syllabus (cont.):

Chap. 7 – Analysis of DESs [2 weeks]
Properties of DESs. Methodologies for the analysis of DESs: the reachability graph and the matricial equation method.

Chap. 8 – DESs and Industrial Automation [1 week]
Relations GRAFCET / Petri networks.
Analysis of industrial automation solutions as DESs.

Chap. 9 – Supervision of Industrial Processes [2 weeks]
Methodologies for supervision. SCADA.
Synthesis based on invariants. Examples of application.

Assessment and grading:

- 2 Preliminary laboratory assignments - training purposes (0% of the final grade).
- 2 Laboratory assignments (20%+20% of the final grade). Groups of 3 students.
- 1 Seminar (20% of the final grade). Topics to be selected with each group.
- Exams (40% of the final grade). Two written.

  Upon student choice, the second exam can be oral.

- Minimum grade: 9.5/20.0 val. in each component.
- ~~Oral discussion for students with grade > 17/20 valores .~~

Extra 1 (one) valor for students attending more than 50% of recitations.

Schedule (suggested)

October 1st 2010

Schedule (according to IST-GOP):

- Recitation classes

Monday	11.00 h – 12.30h	Ea5
Friday	11.00 h – 12.30h	Ea4

- Lab. Classes

Monday	09.30h – 11.00h L1	LSDC4
Friday	09.30h – 11.00h L2	LSDC4

Third session needed?

- Groups register for the Laboratory

Bibliography:

• Automating Manufacturing Systems with PLCs, Hugh Jack (online version available).

• Peterson, James L., "Petri Net Theory and the Modeling of Systems", Prentice-Hall,1981.

•Modeling and Control of Discrete-event Dynamic Systems with Petri Nets and other Tools, Branislav Hruz and MengChu Zhou, 2007. New reference…

--- secondary---

• Programmable Logic Controllers, Frank D. Petruzella, McGraw-Hill, 1996.

• Petri Nets and GRAFCET: Tools for Modeling Discrete Event Systems, R. DAVID, H. ALLA, New York : PRENTICE HALL Editions, 1992.

• Computer Control of Manufacturing Systems, Yoram Koren, McGraw Hill, 1986.

• Cassandras, Christos G., "Discrete Event Systems - Modeling and Performance Analysis", Aksen Associates, 1993.

• Moody, J. e Antsaklis, Supervisory Control of Discrete Event Systems, Kluwer Academic Publishers, 1998.

Industrial Automation

(Automação de Processos Industriais)

Introduction to Automation

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)

Cap. 1 – Introduction to Automation [1 week]

Introduction to components in industrial automation.

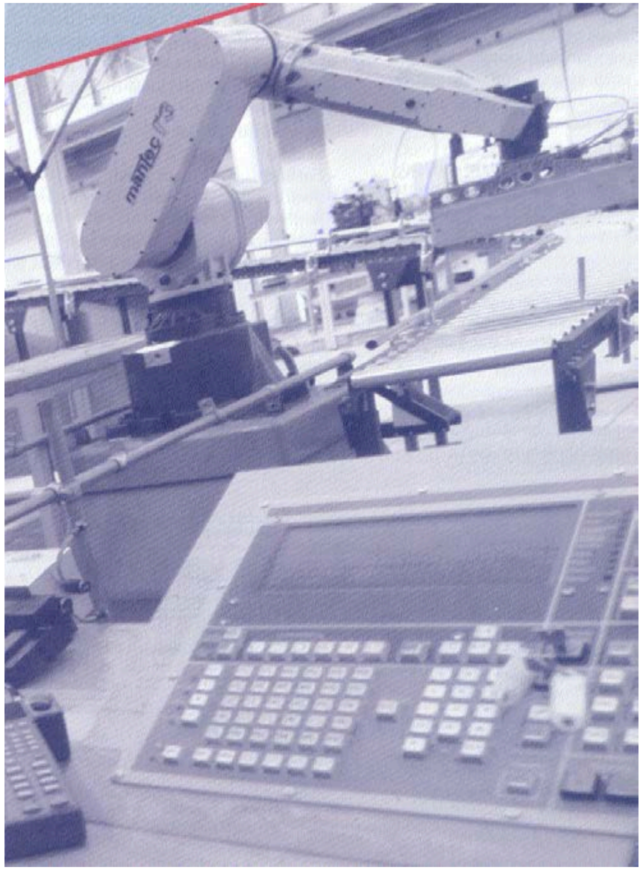
Introduction to methodologies for problem modeling.

Cabled logic versus programmed logic versus networked logic.

Methodologies of work.

Components used in industrial automation

The production of increasing amounts of goods requires the storage and handling of large quantities of resources.

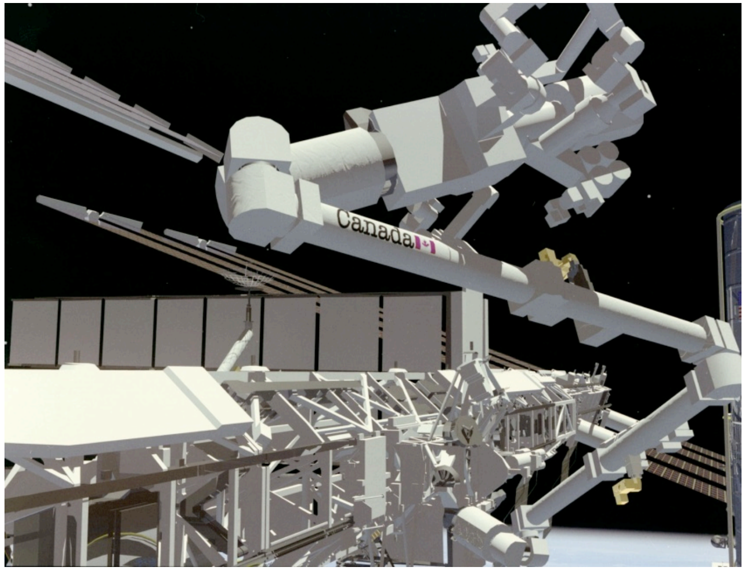


The use of specialized, automatic tools are mandatory.

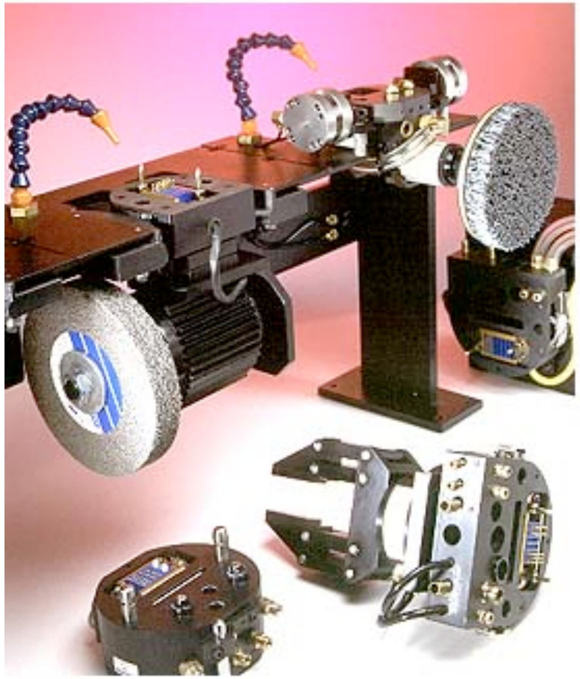
Consistent trend in the last three centuries (since the Industrial Revolution).

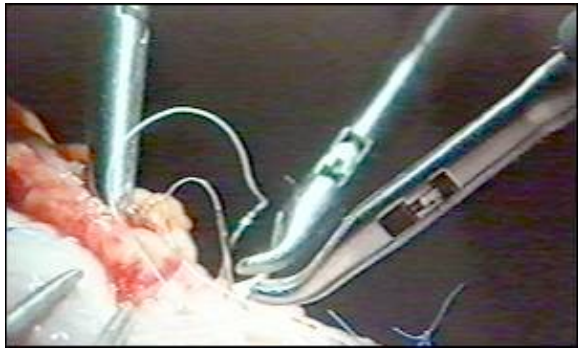
Automation was also fostered by the invention of computers,

Robotic Manipulators



End Effectors





Robotic Manipulators

Major characteristics:

- Number of degrees of freedom
- Types of joints (prismatic/revolution/...)
- Programming tools and environments (high level languages, teach pendent, ...)
- Workspace
- Accuracy, fiability
- Payload and robustness

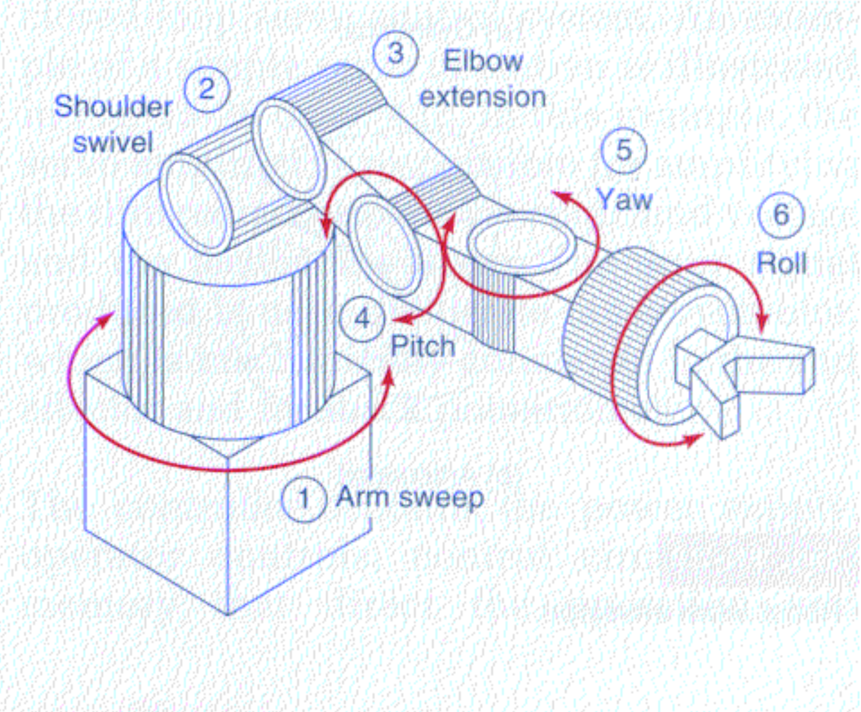


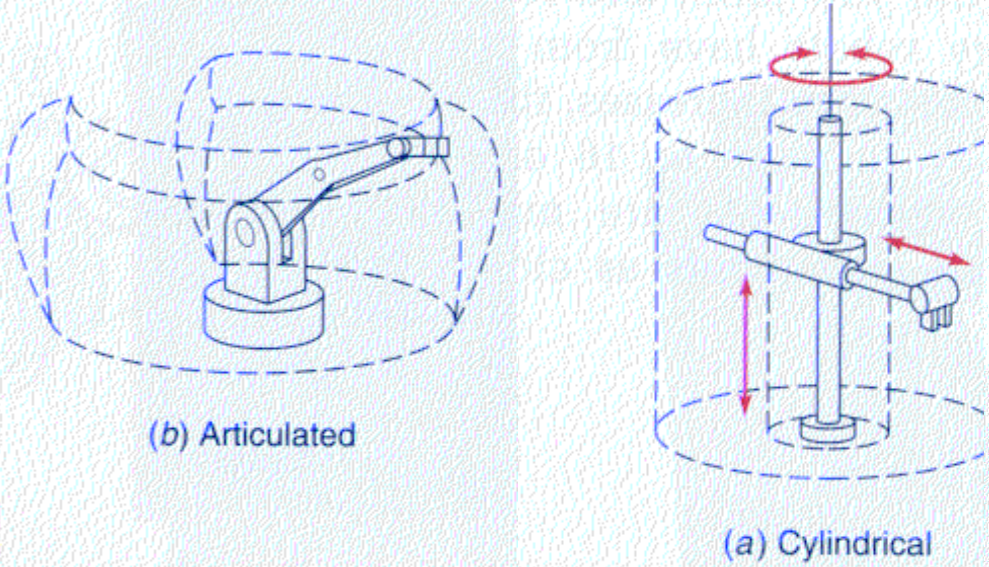
Fig. 15-22

Six-axis robot arm.

Robotic Manipulators

Workspace:

Examples

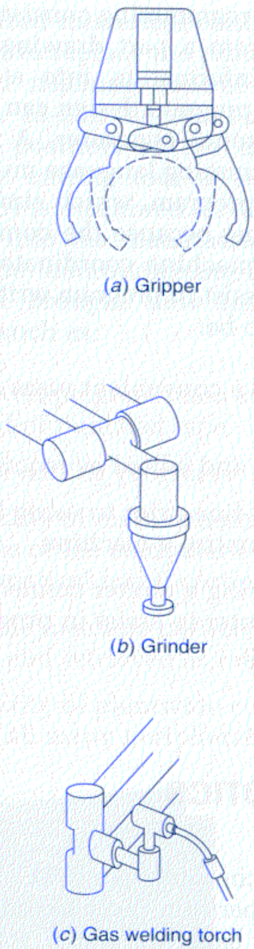


(b) Articulated

(a) Cylindrical

Fig. 15-23

Robot work envelope.



(a) Gripper

(b) Grinder

(c) Gas welding torch

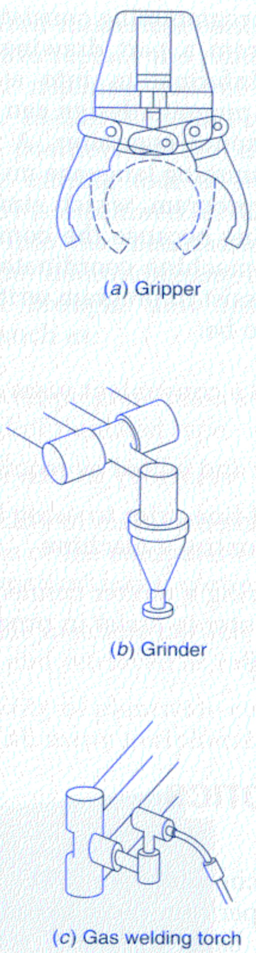
Fig. 15-24

End-of-arm tooling devices.

Robotic Manipulators

Central problems to adress and solve:

- Direct kinematics
- Inverse Kinematics
- Trajectory generation
- Coordinate frames where tasks are specified
- Level of abstraction of the programming languages


(*a*) Gripper

(*b*) Grinder

(*c*) Gas welding torch

Fig. 15-24
End-of-arm tooling devices.

Robotic Manipulators

Use in Flexible

Cells of Fabrication:

it is required that the manipulators have correct interfaces for the synchonization and inputs for external commands.

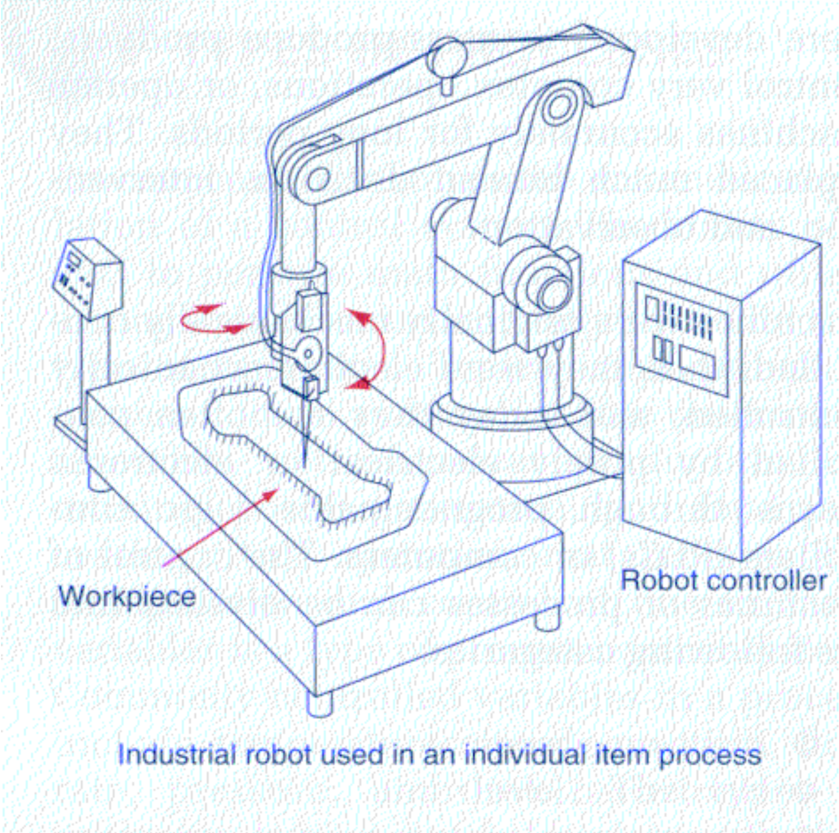


Fig. 14-3
Individual product production.

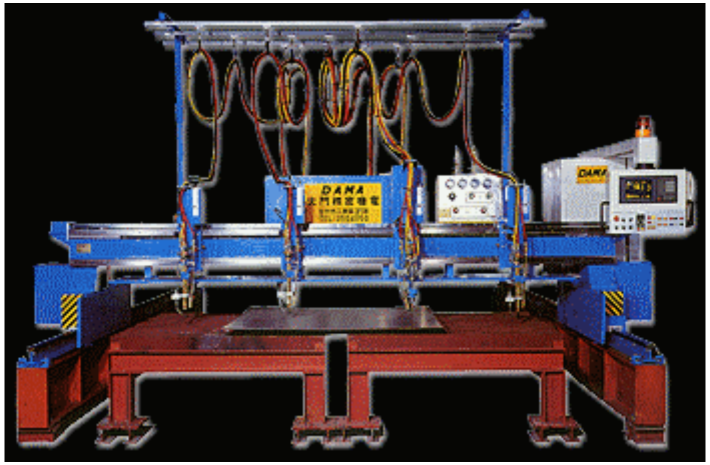
Computerized Numerical Controlled Machines

Major characteristics:

- Number of degrees of freedom
- Interpolation methods
- Load/unload automation, and also in tool change
- Programming (high level languages, teach pendent, ...)
- Workspace
- Accuracy, reliability
- Payload and robustness
- Interface
- Synchronization with exterior

Examples:
Milling, Lathes, ...

Computerized Numerical Controlled Machines

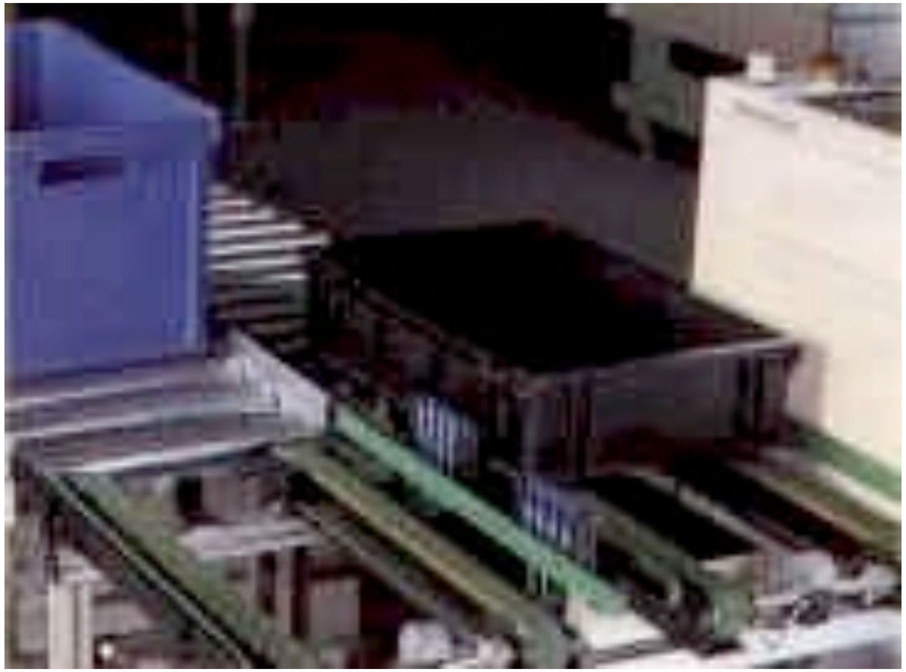


Solutions for Handling materials

For transport...

Major characteristics:

- Load/unload automation
- Accuracy, reliability
- Payload and robustness
- Interface
- Synchronization with exterior



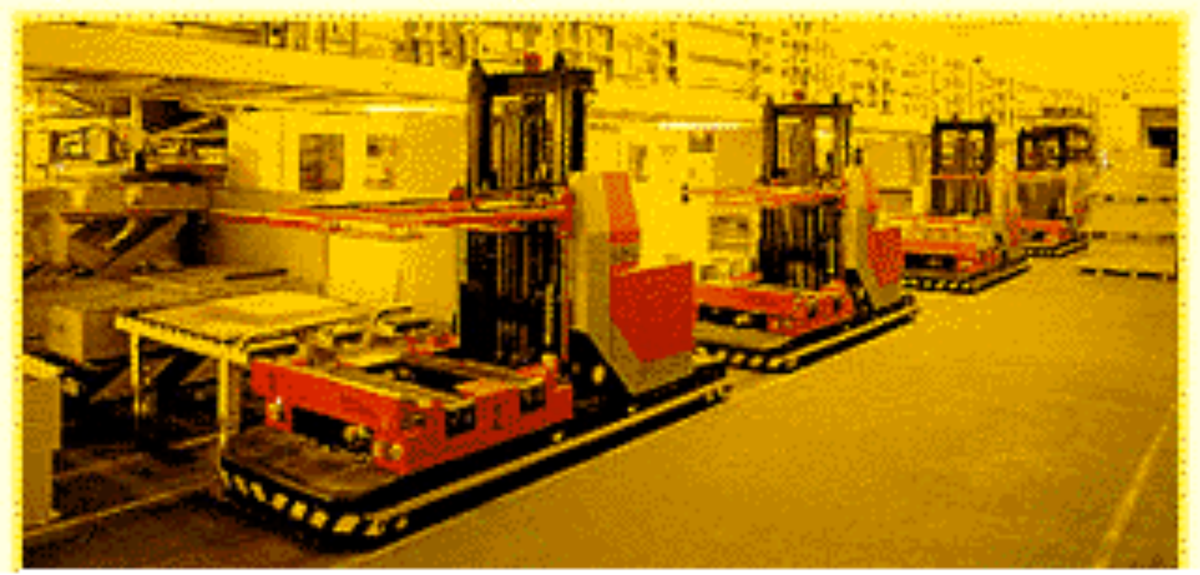
AGVs (Automatic Guided Vehicles)

Major characteristics:

- Load/unload automation
- Accuracy, reliability
- Payload and robustness
- Interface
- Synchronization with exterior

AGVs (Automatic Guided Vehicles)

Example of fleet operating in industry



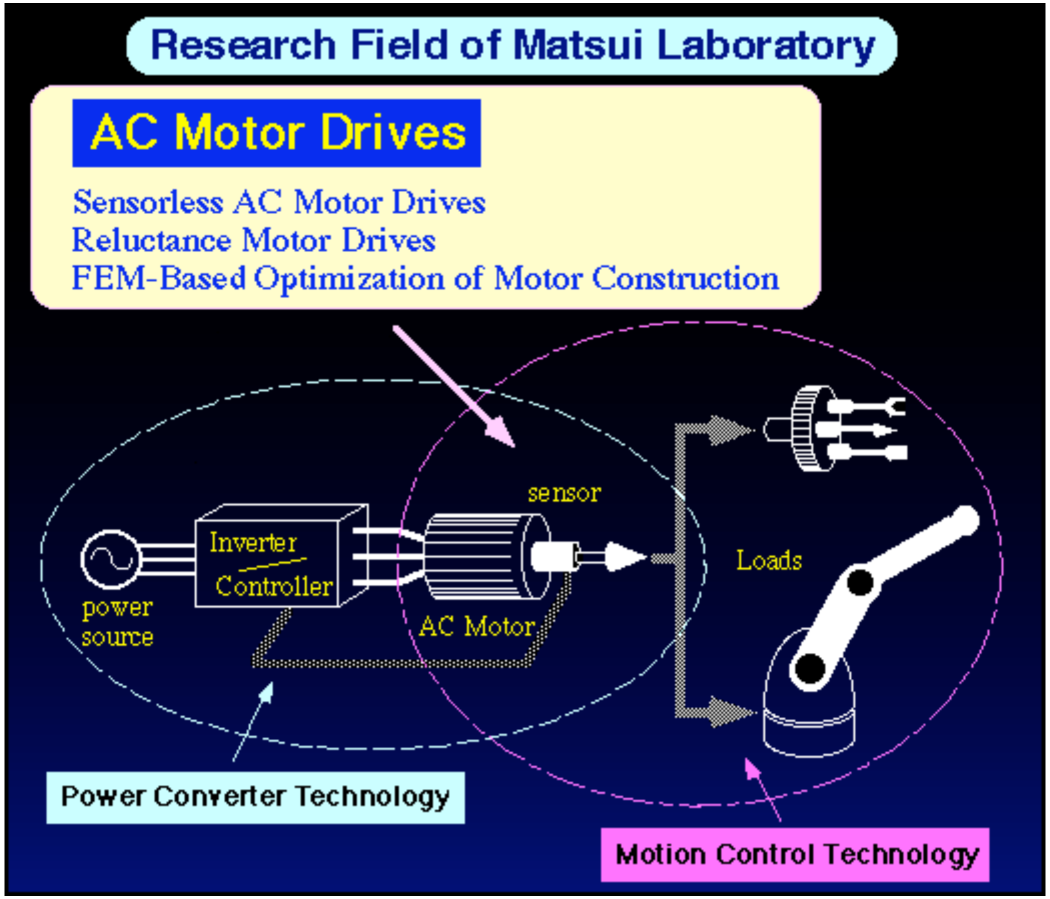
Actuation

Motors

Major características:

- Tipe of start
- Tipe of control
- Accuracy, reliability
- Payload and robustness
- Interface with exterior
- Synchronization

Exemple of AC motor, with driver



Research Field of Matsui Laboratory

AC Motor Drives

Sensorless AC Motor Drives
Reluctance Motor Drives
FEM-Based Optimization of Motor Construction

Power Converter Technology

Motion Control Technology

Specific Components

Factury example: production of aluminium packs

Cabled Logic versus ...

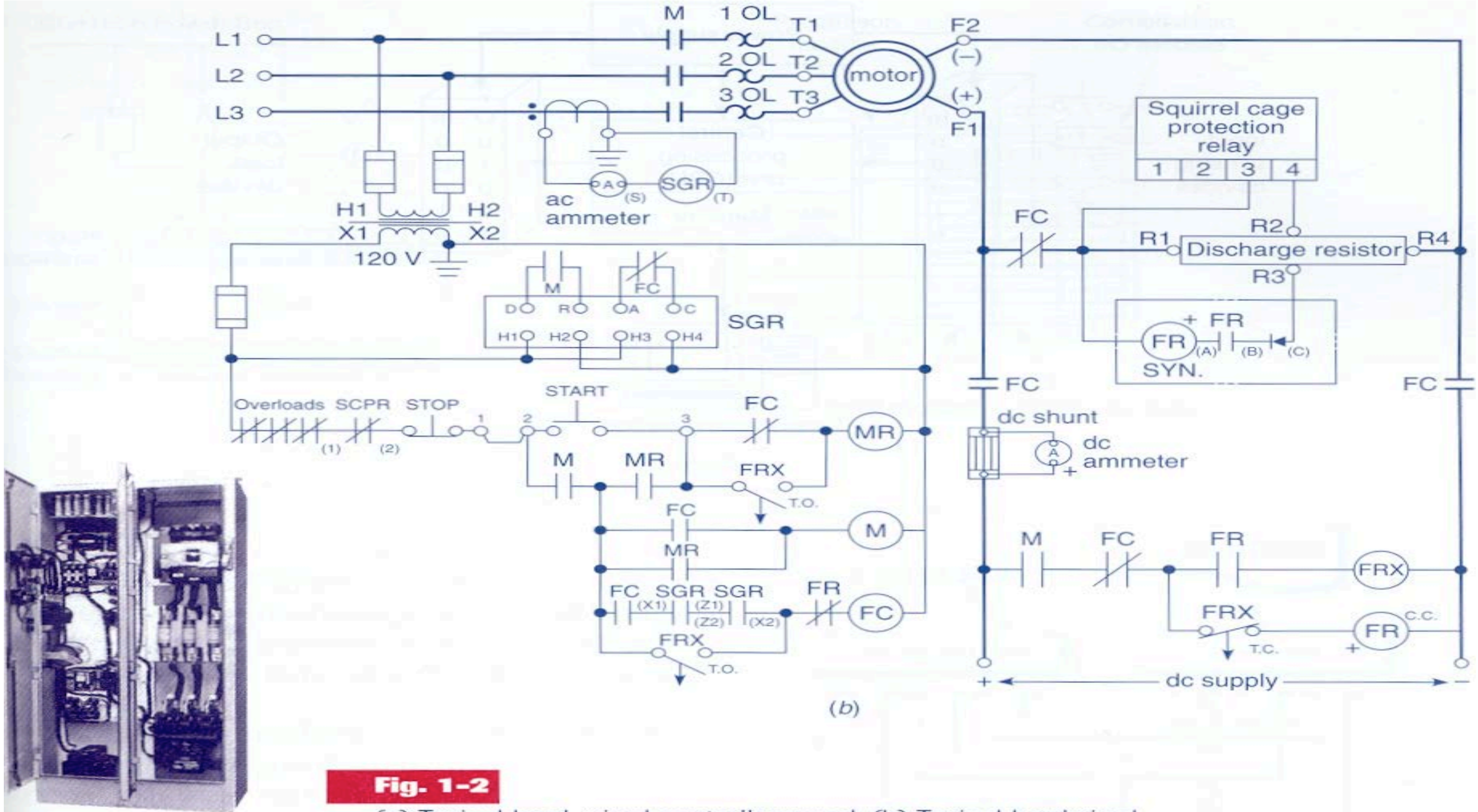
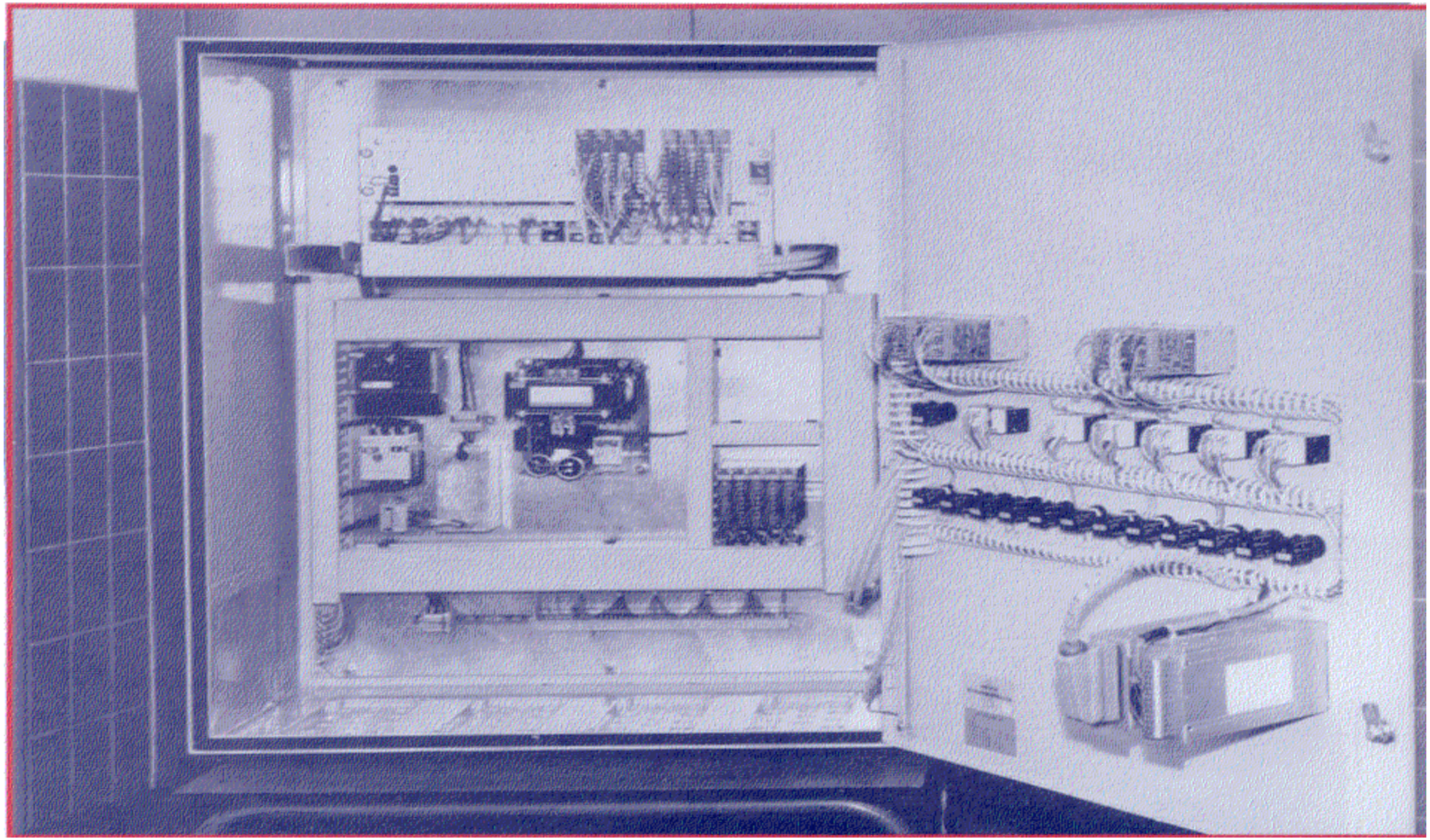


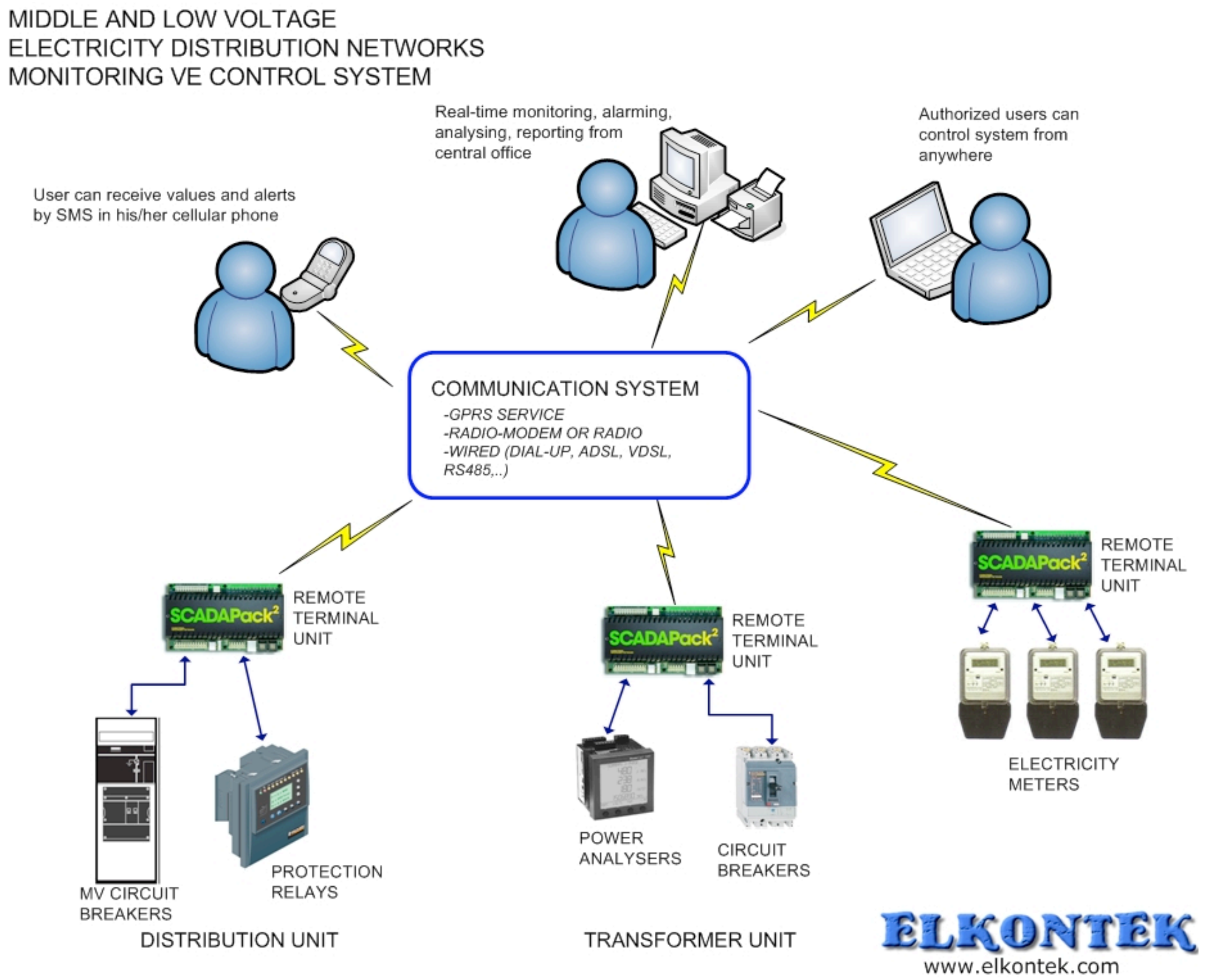
Fig. 1-2

(a) Typical hard-wired controller panel. (b) Typical hardwired diagram. (Courtesy of Allen-Bradley Company, Inc.)

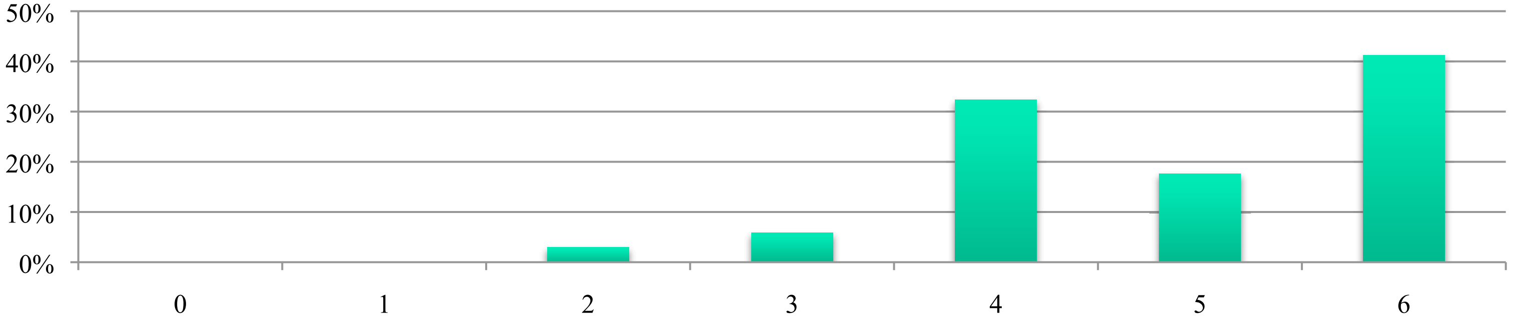
... versus Programmed Logic …

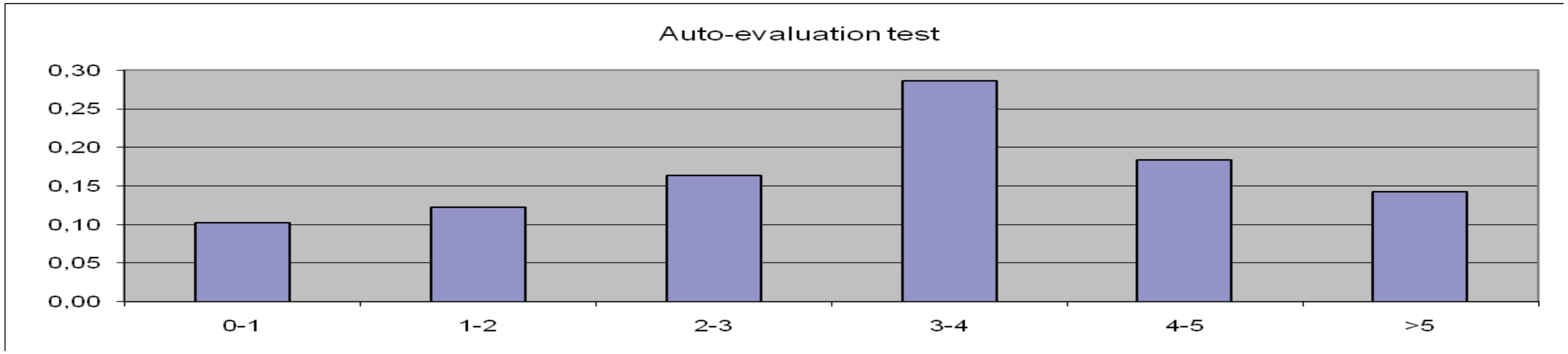


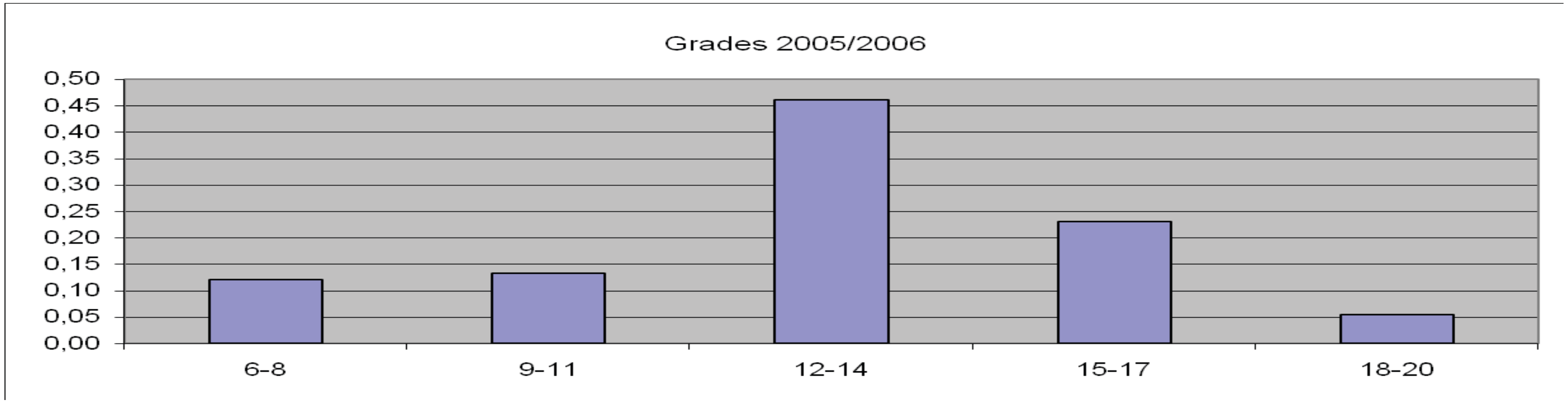
... versus Networked Logic



Analysis of the auto-evaluation test







Introduction to methodologies

for problem modeling

in

Industrial Automation

Solenoide Valve

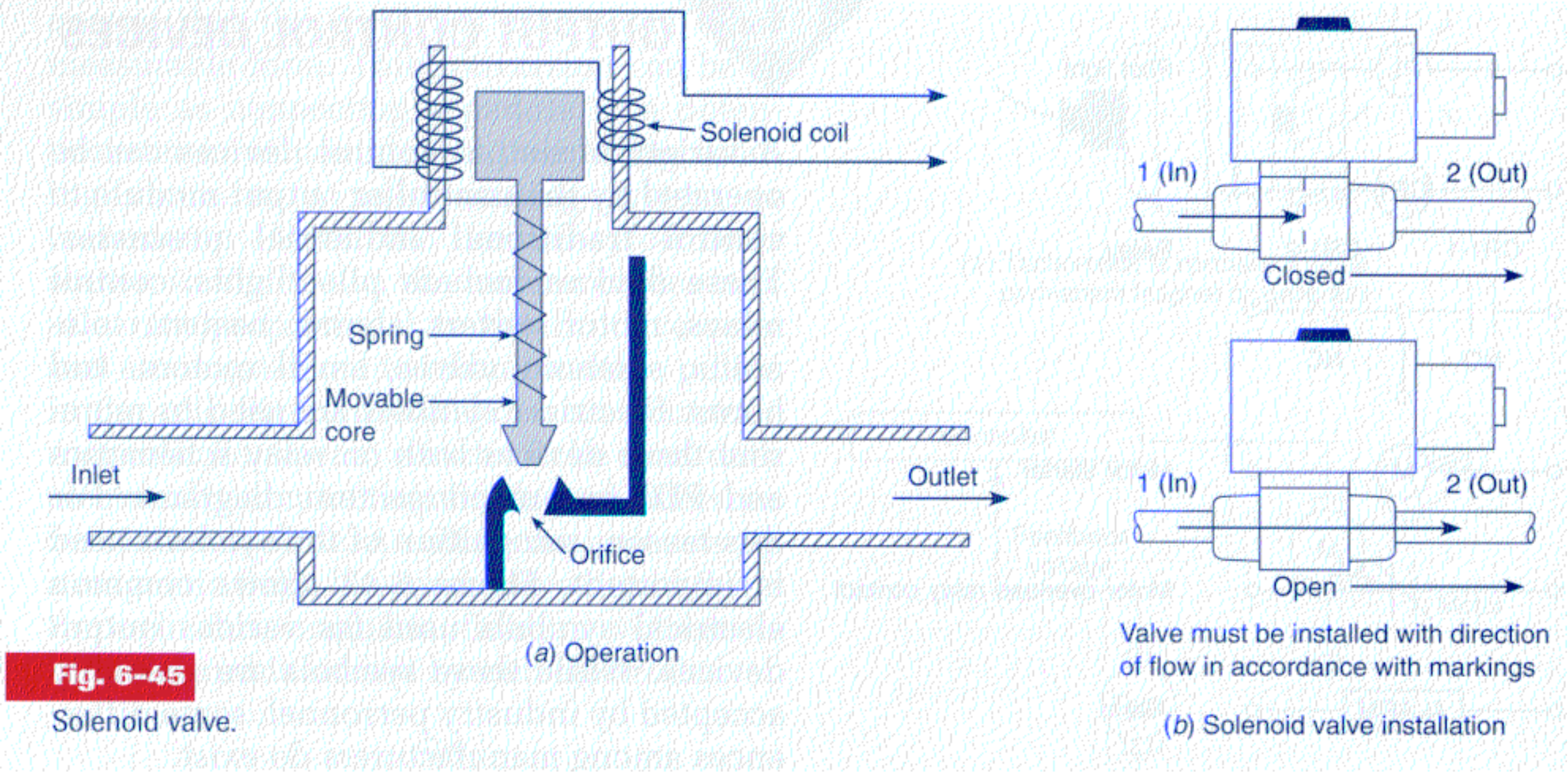
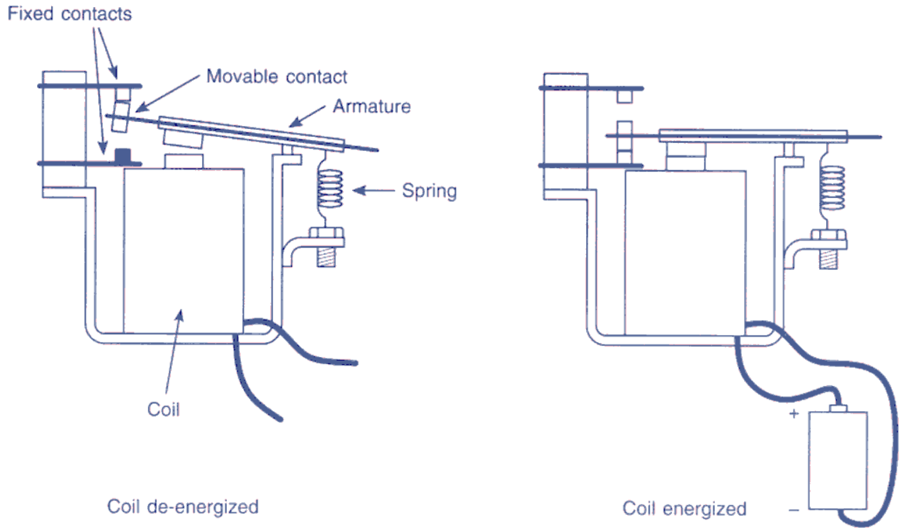

Solenoid coil
Spring
Movable core
Inlet
Outlet
Orifice
(a) Operation
1 (In)
2 (Out)
Closed
1 (In)
2 (Out)
Open
Valve must be installed with direction of flow in accordance with markings
(b) Solenoid valve installation

Fig. 6-45

Solenoid valve.

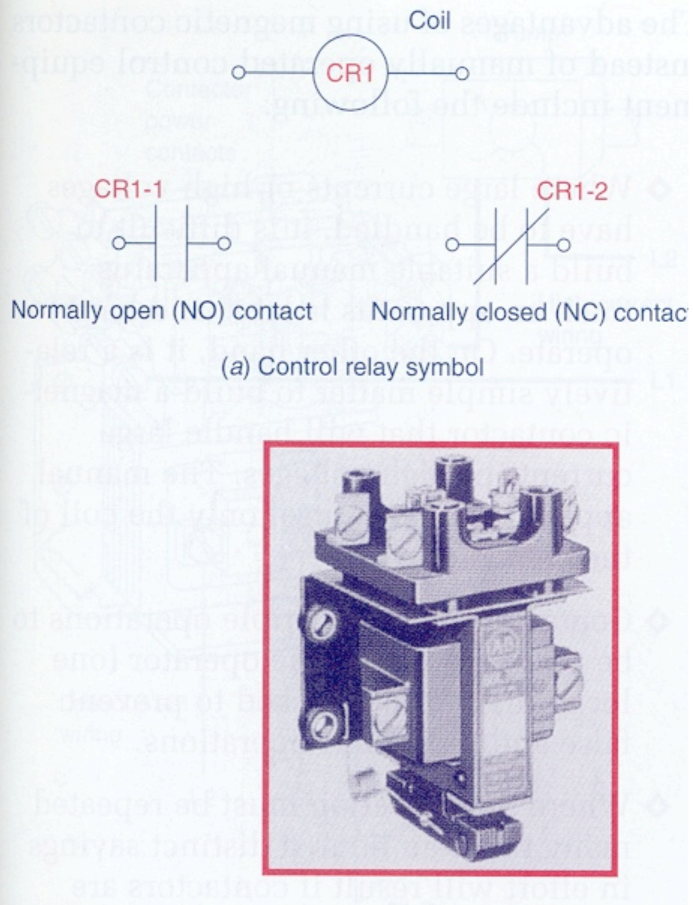
Command Relay



Coil de-energized

Coil energized

Fig. 6-1
Electromagnetic control relay operation.



Normally open (NO) contact

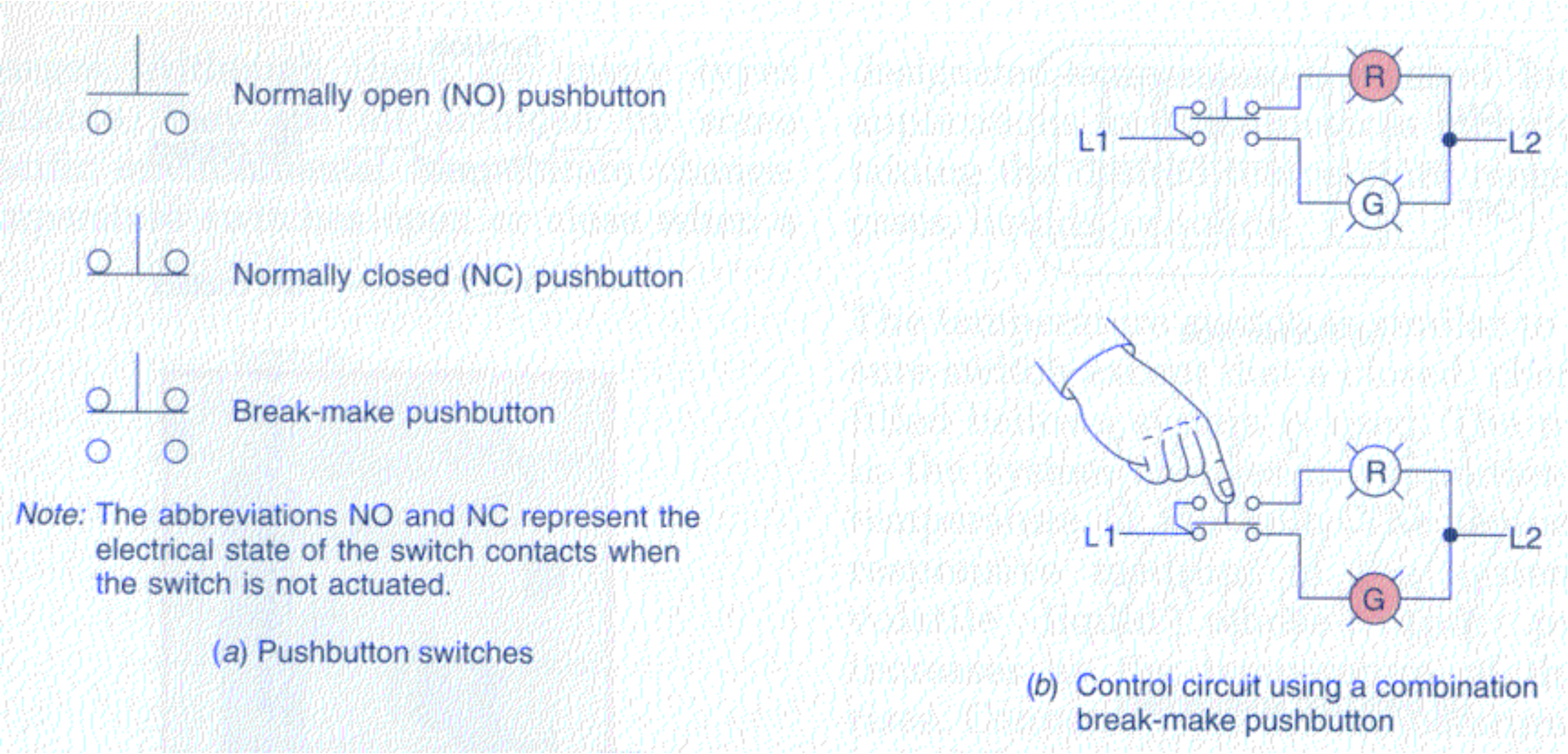
Normally closed (NC) contac

(*a*) Control relay symbol

(*b*) Typical industrial control relay. *(Courtesy of Allen-Bradley Company, Inc.)*

Fig. 6-2
Control relay.

Push buttons

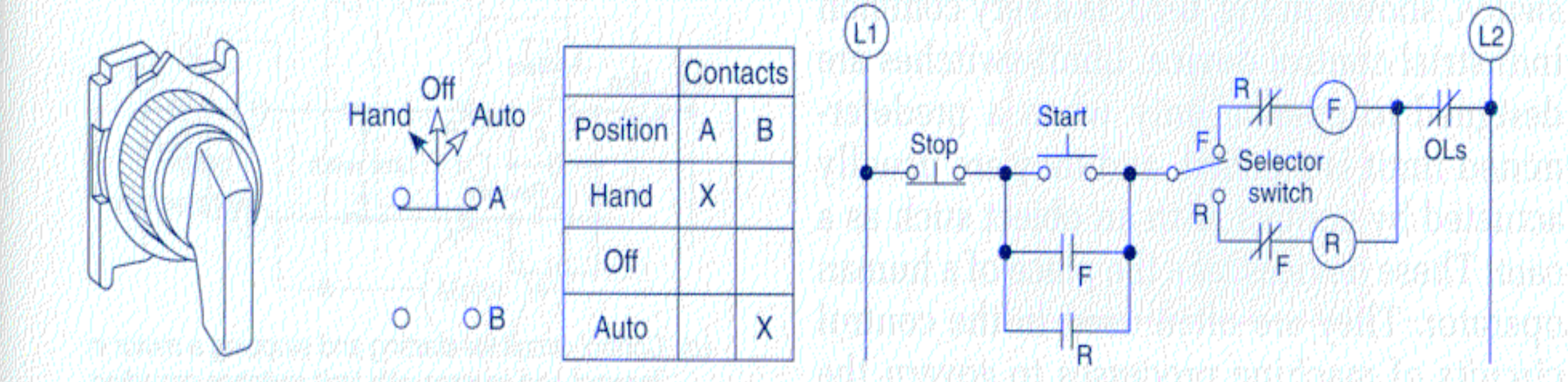

Normally open (NO) pushbutton
Normally closed (NC) pushbutton
Break-make pushbutton
L1
R
G
L2
L1
R
G
L2

Note: The abbreviations NO and NC represent the electrical state of the switch contacts when the switch is not actuated.

(*a*) Pushbutton switches

(*b*) Control circuit using a combination break-make pushbutton

Selector with three positions



	Contacts	
Position	A	B
Hand	X	
Off		
Auto		X

(*a*) Selector switch operator

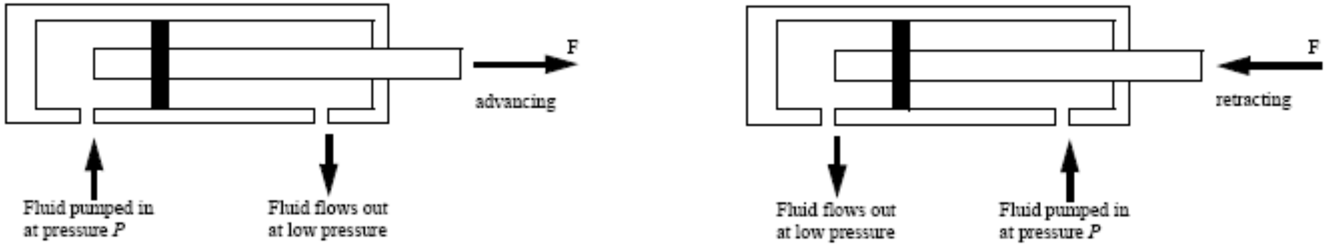
(*b*) Three-position selector switch and truth table

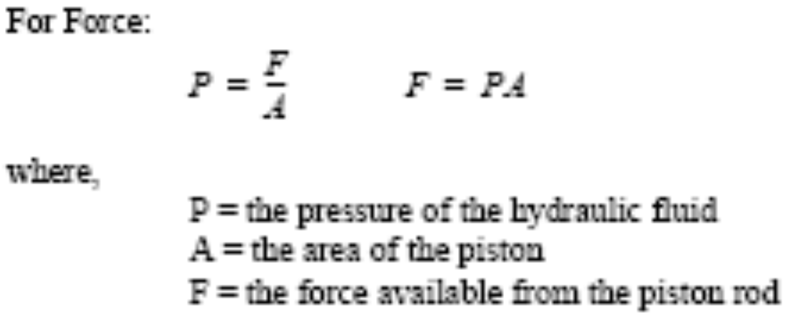
(*c*) Selector switch used in conjunction with a reversing motor starter to select forward or reverse operation of the motor

Fig. 6-11

Selector switch.

Cylinders (Pneumatics)


F
advancing
Fluid pumped in at pressure P
Fluid flows out at low pressure
F
retracting
Fluid flows out at low pressure
Fluid pumped in at pressure P



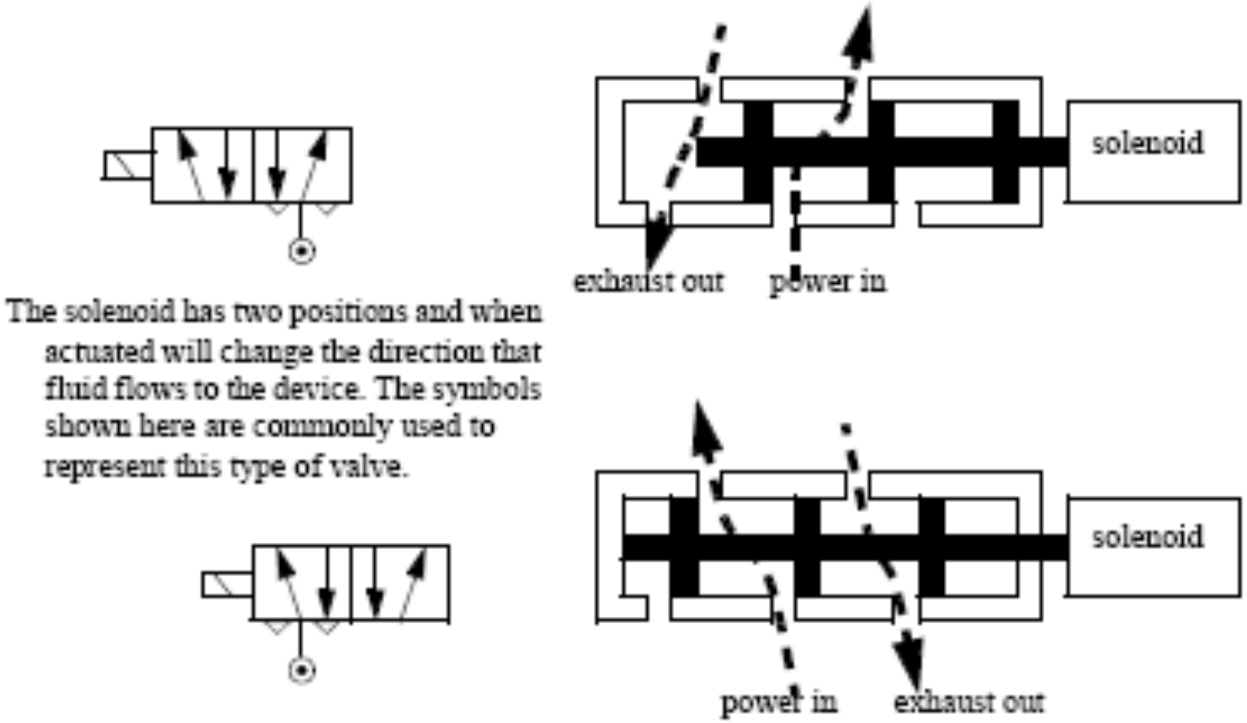
For Force:

$$P = \frac{F}{A} \qquad F = PA$$

where,

P = the pressure of the hydraulic fluid
A = the area of the piston
F = the force available from the piston rod

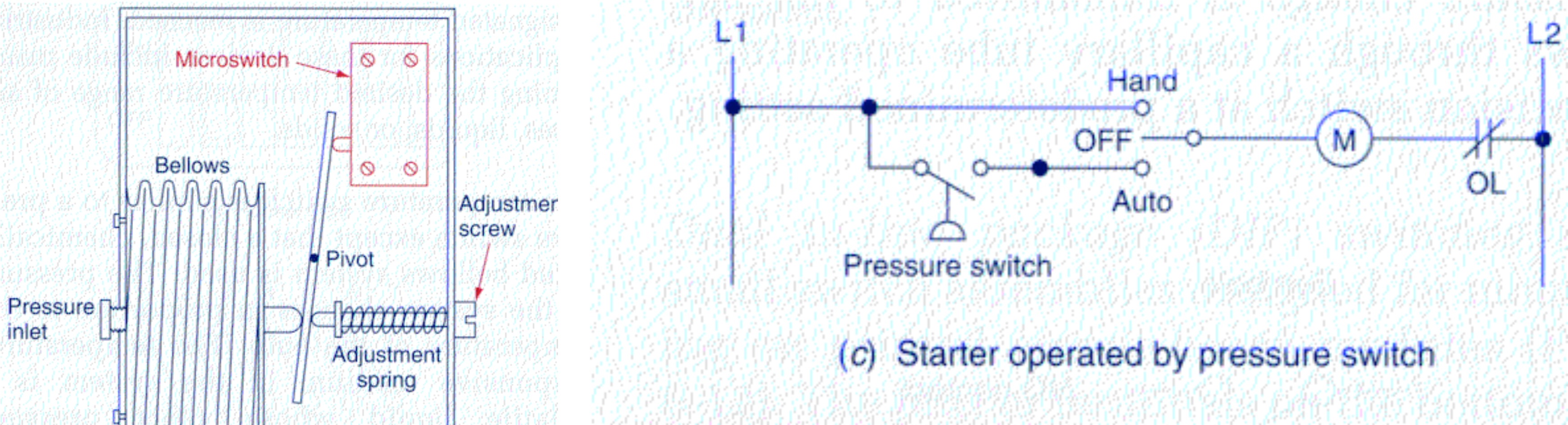
Valves(Electro-pneumatics)



The solenoid has two positions and when actuated will change the direction that fluid flows to the device. The symbols shown here are commonly used to represent this type of valve.

Sensors

Pressure Switch

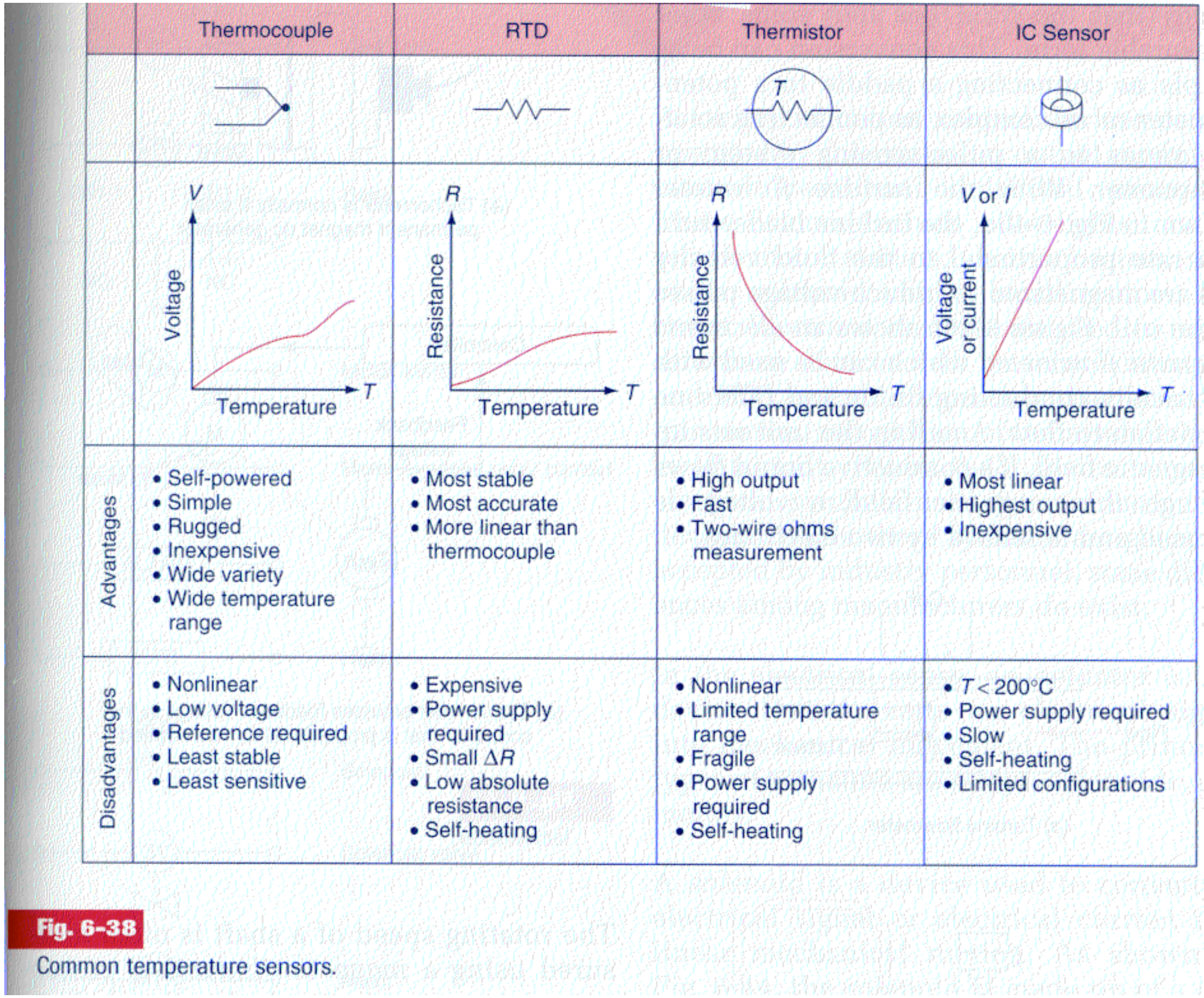


(*b*) Bellows

(*c*) Starter operated by pressure switch

Fig. 6-15 (continued) Pressure switch.

Temperature Sensors


V
Voltage
Temperature
T
R
Resistance
Temperature
T
R
Resistance
Temperature
T
V or I
Voltage or current
Temperature
T

	Thermocouple	RTD	Thermistor	IC Sensor
Advantages	• Self-powered • Simple • Rugged • Inexpensive • Wide variety • Wide temperature range	• Most stable • Most accurate • More linear than thermocouple	• High output • Fast • Two-wire ohms measurement	• Most linear • Highest output • Inexpensive
Disadvantages	• Nonlinear • Low voltage • Reference required • Least stable • Least sensitive	• Expensive • Power supply required • Small ΔR • Low absolute resistance • Self-heating	• Nonlinear • Limited temperature range • Fragile • Power supply required • Self-heating	• $T < 200°C$ • Power supply required • Slow • Self-heating • Limited configurations

Fig. 6-38

Common temperature sensors.

Termocouple

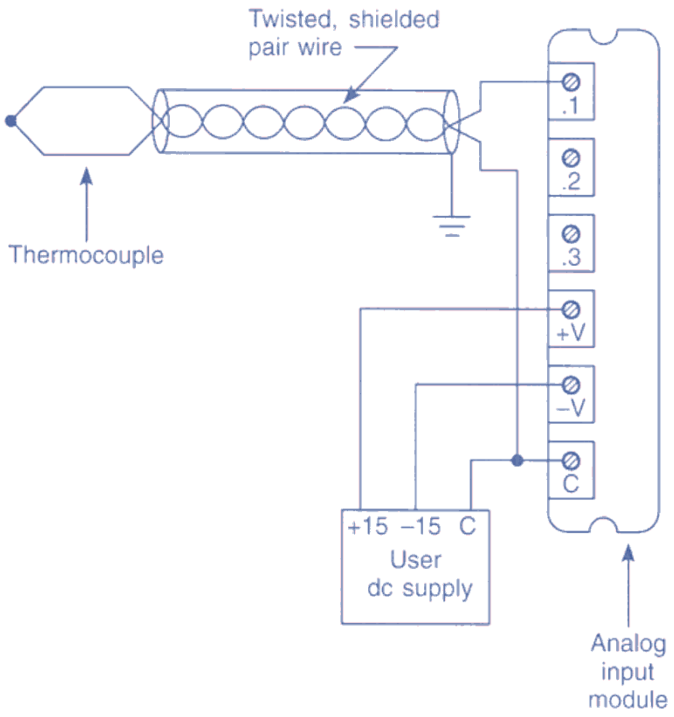
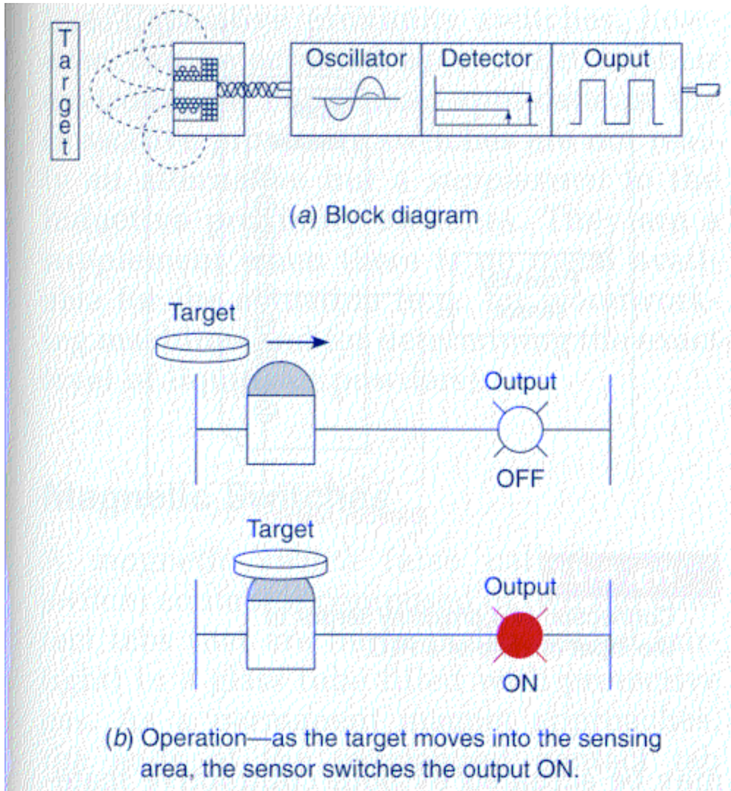


Fig. 2-12

Typical thermocouple connection to an analog input module.

Proximity detector



(*b*) Operation—as the target moves into the sensing area, the sensor switches the output ON.

Fig. 6-20

Inductive proximity sensor.

Magnetic detector

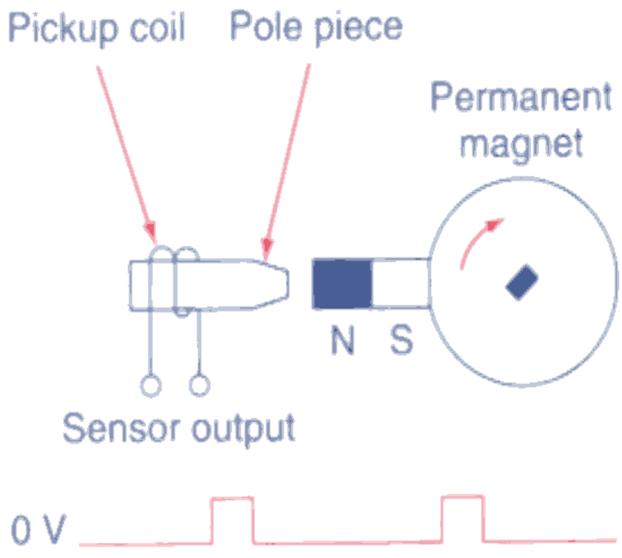


Fig. 6-42
Magnetic pickup sensor.

Magnetic switch

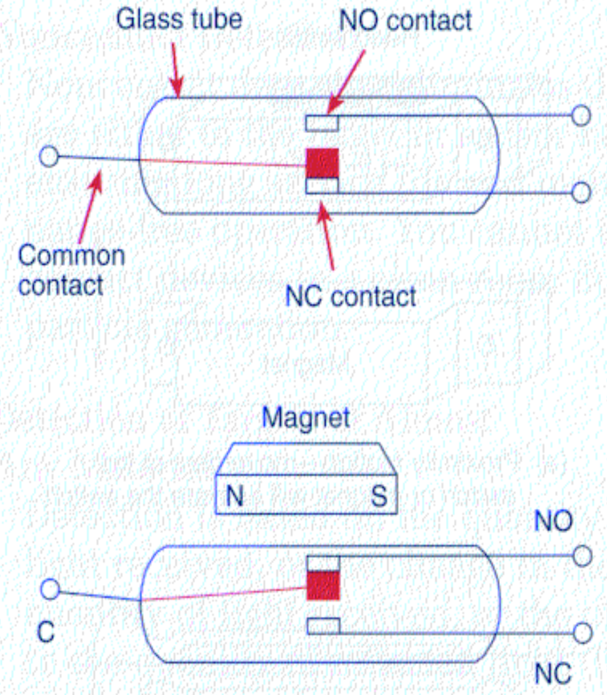
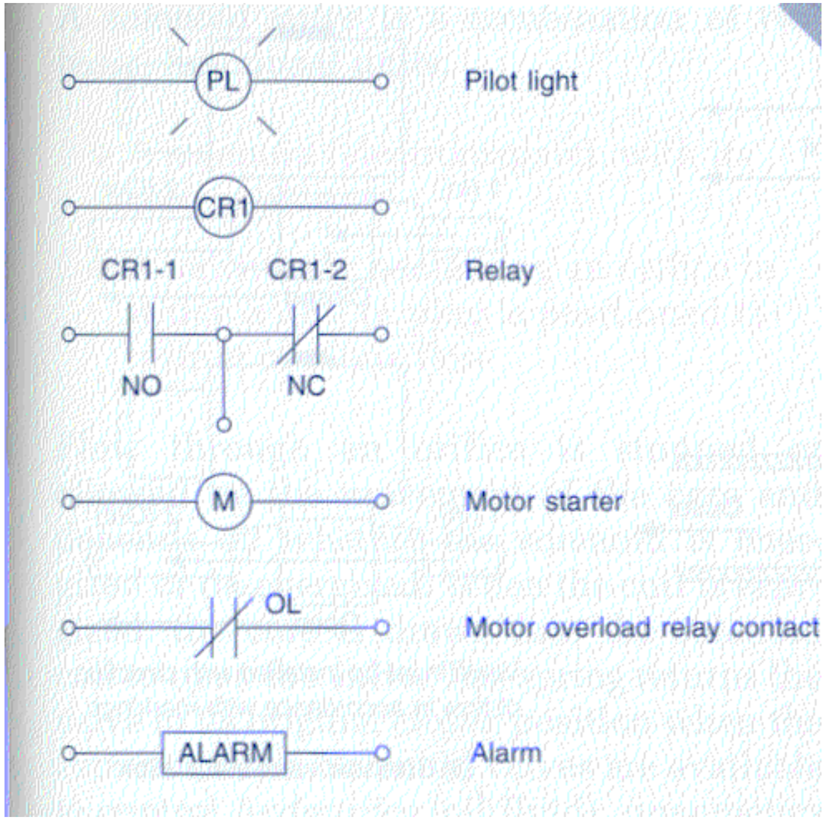


Fig. 6-25
Magnetic switch (reed switch).

Symbols associated to all components

Standards



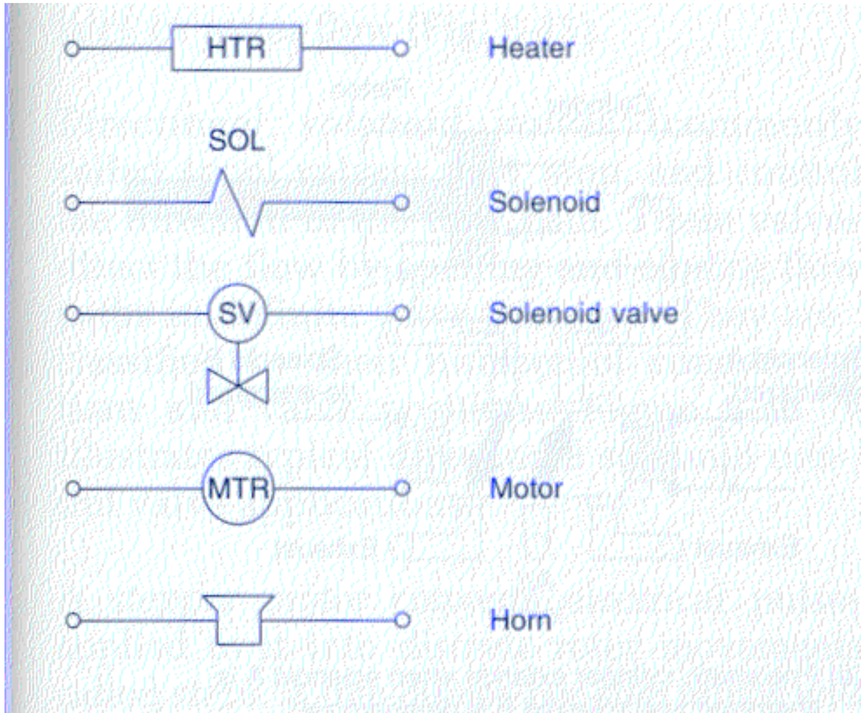


Fig. 6-43

Symbols for output control devices.

Ladder Diagram

Or

Contact Diagram

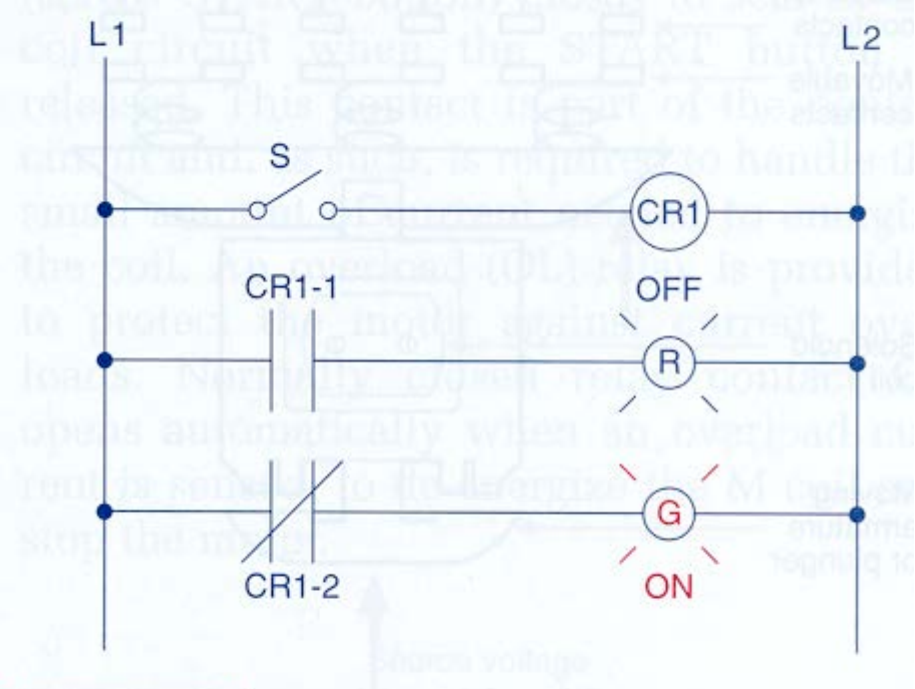


Fig. 6-3

Relay circuit–switch open.

Methodologies for the implementation of solutions in industrial automation

Contacts diagram

Example

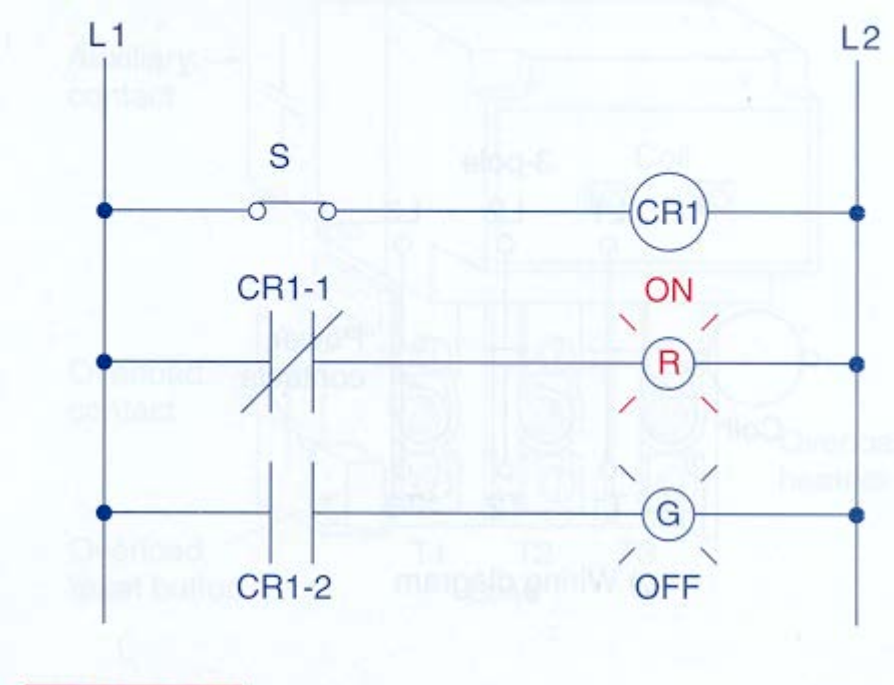


Fig. 6-4

Relay circuit–switch closed.

Example:

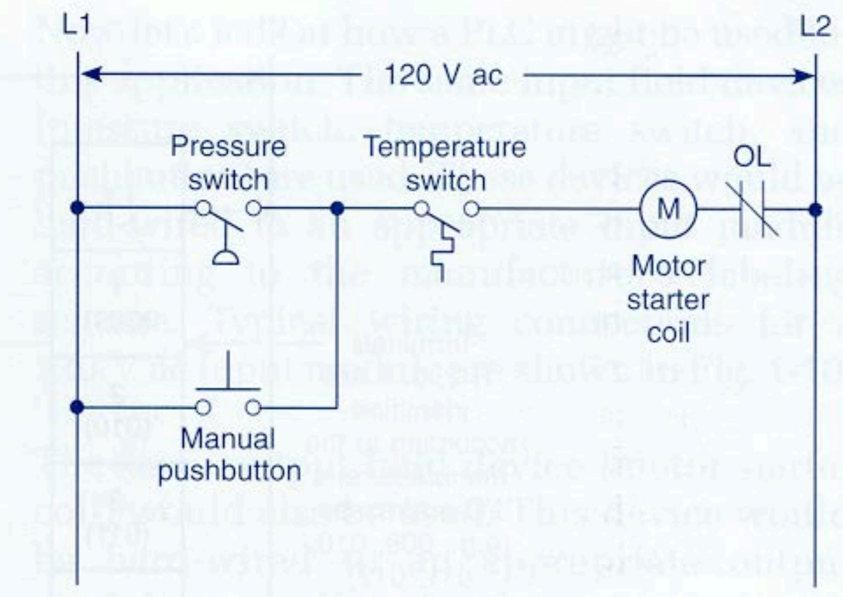


Fig. 1-13 Relay ladder diagram for modified process.

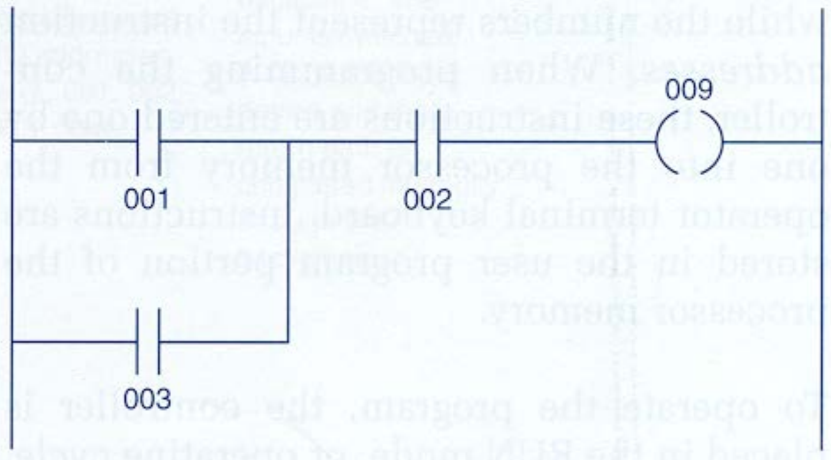
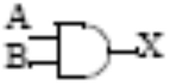


Fig. 1-14 PLC ladder logic diagram for modified process.

Logic Functions

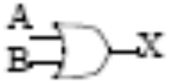
AND



$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

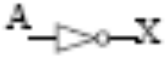
OR



$X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

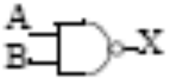
NOT



$X = \bar{A}$

A	X
0	1
1	0

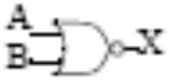
NAND



$X = \overline{A \cdot B}$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

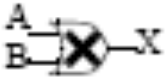
NOR



$X = \overline{A + B}$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

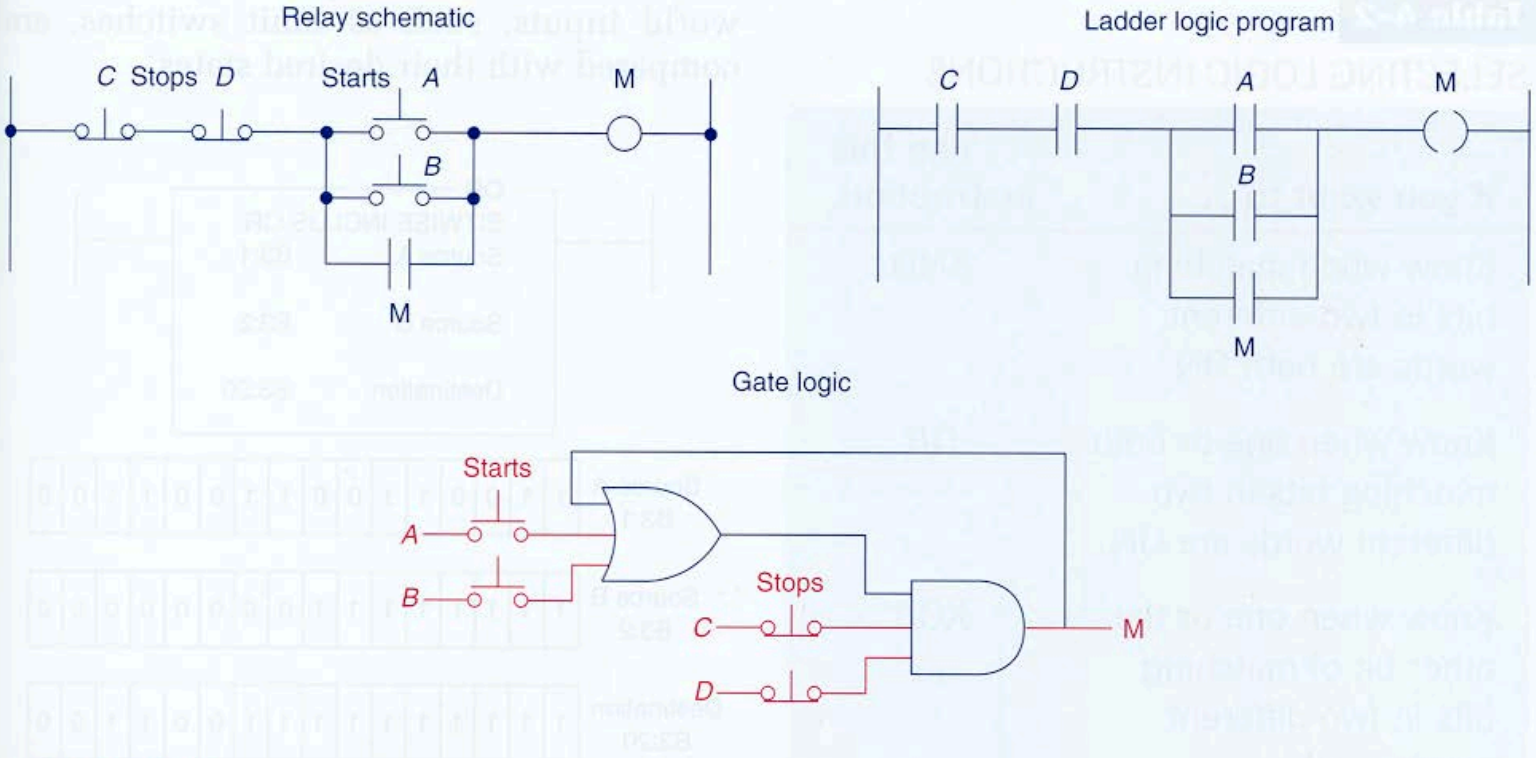
EOR



$X = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

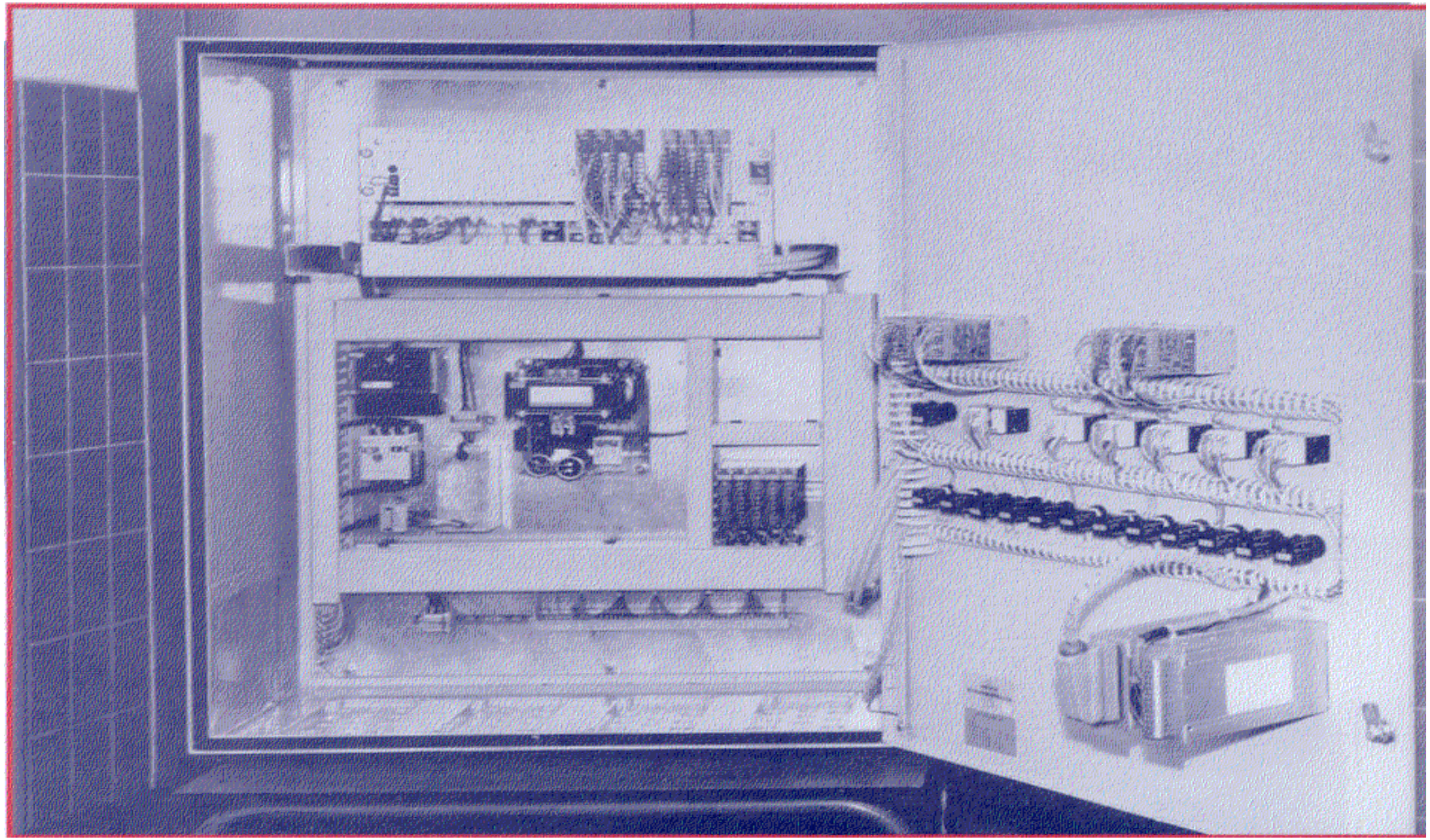
Example:



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.

To exploit the advantages of Programmed Logic



Industrial Automation

(Automação de Processos Industriais)

Introduction to PLCs

http://www.isr.ist.utl.pt/~pjcro/courses/api0910/api0910.html

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

Syllabus:

Chap. 1 – Introduction to Automation [1 week]

...

Chap. 2 – Introduction to PLCs [2 weeks]

Components of Programmable Logic Controllers (PLCs).
Internal architecture and functional structure.
Input / output interfaces. Interconnection of PLCs .

...

Chap. 3 – PLCs Programming Languages [2 weeks]

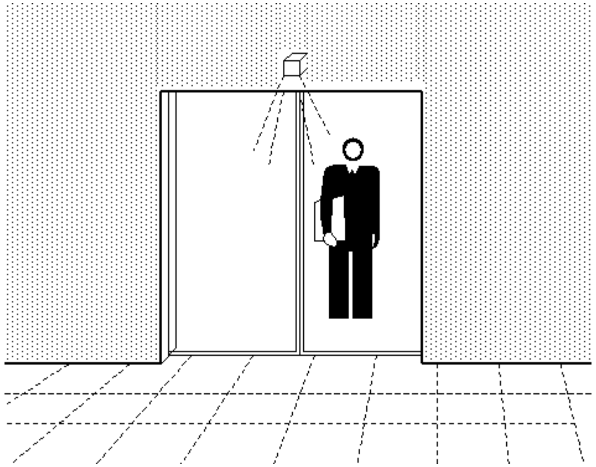
Some resorces available online on PLCs

History :	http://www.plcs.net/chapters/history2.htm
Tutorial:	http://www.koldwater.com/downloadform.htm http://www.htservices.com/Tutorials/plctutorial1.htm http://www.sea.siemens.com/step/templates/lesson.mason?plcs:1:1:1
Simulators:	http://www.thelearningpit.com/psim/psim.html http://www.keyence.com/plc/kvl.htm http://www.autoware.com/english/demo.htm http://www.apalmertraining.com/download.htm http://tytang.hypermart.net/cgi-bin/frame.pl?file=PLC_sim/index.html http://www.thelearningpit.com/psim/psim.html
Bibliography :	Automatic Manufacturing Systems with PLCs, Hugh Jack (online version available) Programming Logic Controller s, Frank D. Petruzella ...
Standards:	http://www.plcopen.org/

An Automation Example
Solution based on PLCs

Example:

Automation of the Main Entrance Door, in *"PLCs Theory,"* ***[Omron]***



Example:

Automation of the Main Entrance Door, in *"PLCs Theory,"* ***[Omron]***

Functional Specifications

An automatic system that could command the oppening and close of a door is the main purpose of these specifications.

The command operation will be automatic and manual. There must be a selector with two positions in a front pannel of command to select the mode of operation.

The manual mode resorts to the use of two push buttons to open and close the door. Once the OPEN push button is pressed, the door will be openned until the operation is completed, as detected by a limit switch. Upon pushing the CLOSE button the door will be commanded to close , untill the end of the operation is detected by other limit switch.

The automatic mode of operation resorts to the use of two sensors, that detect the proximity of the users. When a person is detected the automatic opening of the door starts. The door remais openned for a period from 5 to 20 seconds, following the null detection if the user. After that period the door starts to close. If during this last phase the presence of other user is detected the close operation is aborted and a new cycle of opening starts.

Example:

Automation of the Main Entrance Door, in *“PLCs Theory,”* ***[Omron]***

Technological Specifications

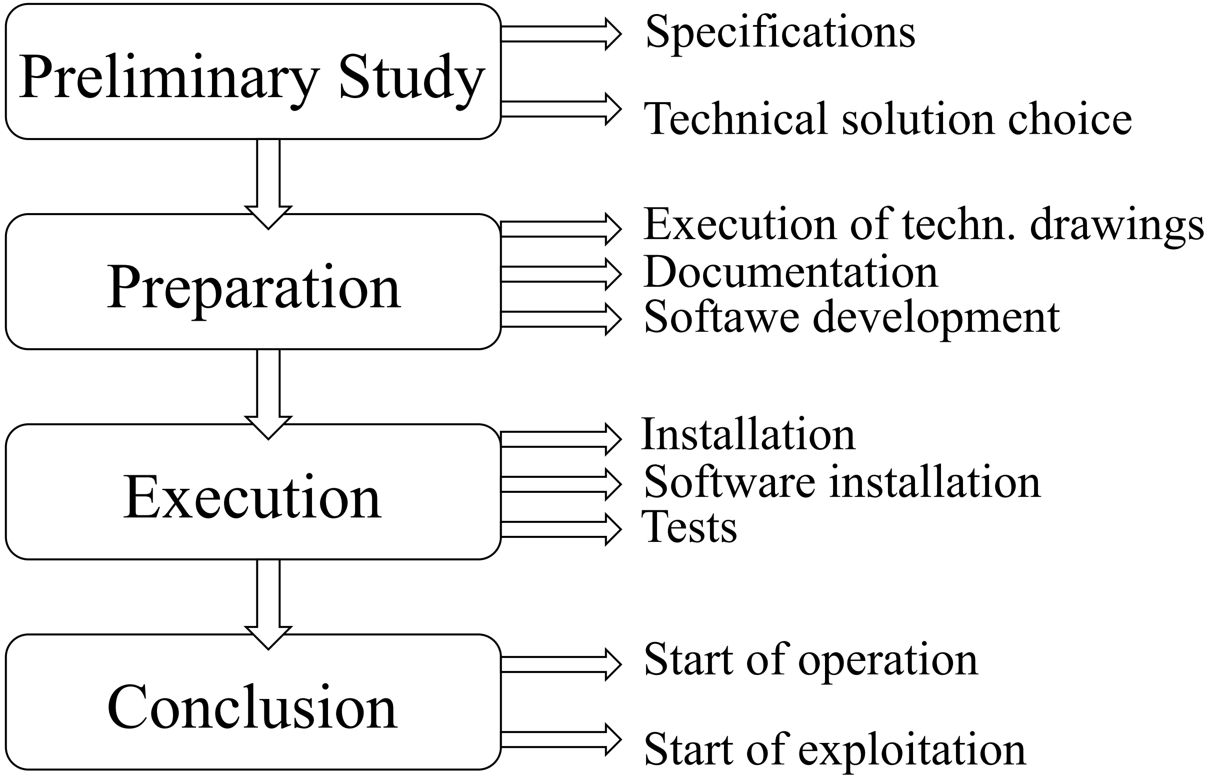
The proximity sensor that detects the users must be of a model that can be installed over the door (one in the interior and other in the exterior), and must be based on the reflection of infrared radiations, with output by transistor. The sensor sensivity must be tunned such taht its output becomes active if an user is at 2 meters of distance or less.

The motor that activates the open and close of the door must be electrical , three-phasic, ..., etc.

Operating Specifications

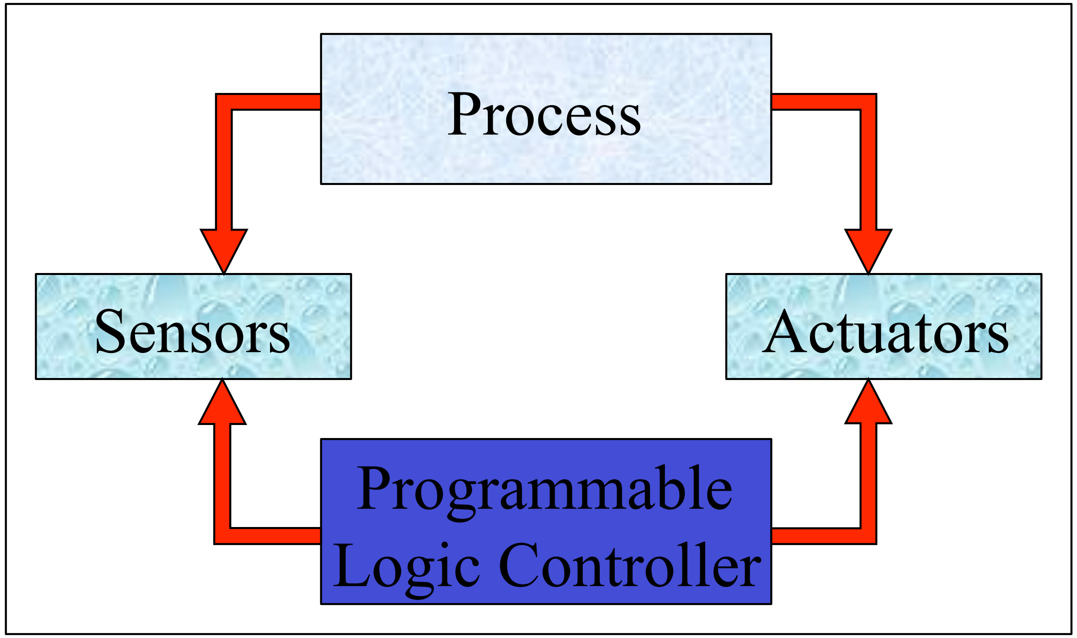
A key must be required to be used in the model of the automatic-manual selector. A counter of the number of operations should be incorporated in the solution, to identify when maintenance is required. The maintenance must be at each 10000 operations, ... etc

Phases of a Project in EE&CS: (Automation included)



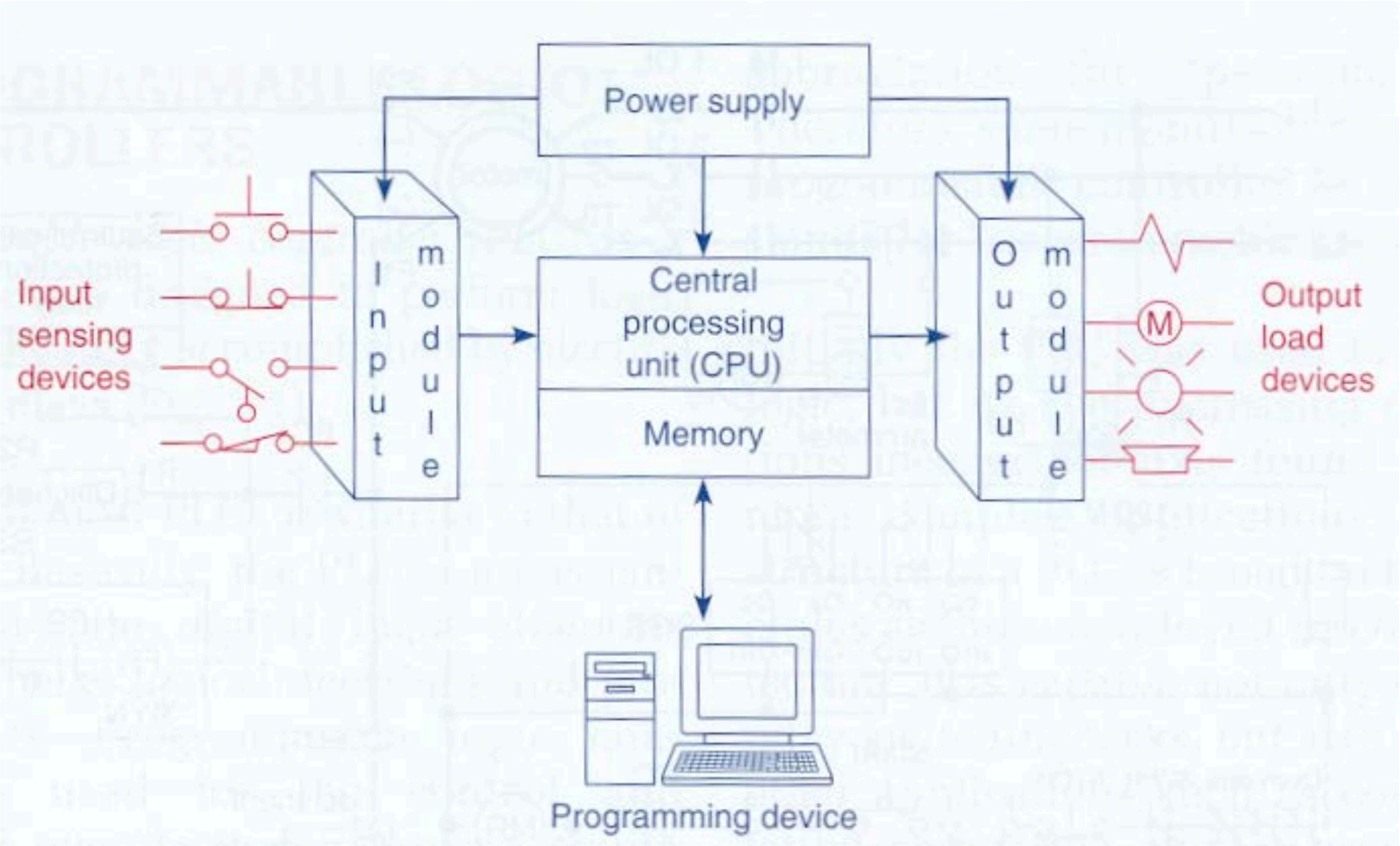
Automation Problems
PLC based solutions

To use PLCS the connection to input devices (for detection and sensing) and to output devices (for command and control) is required.



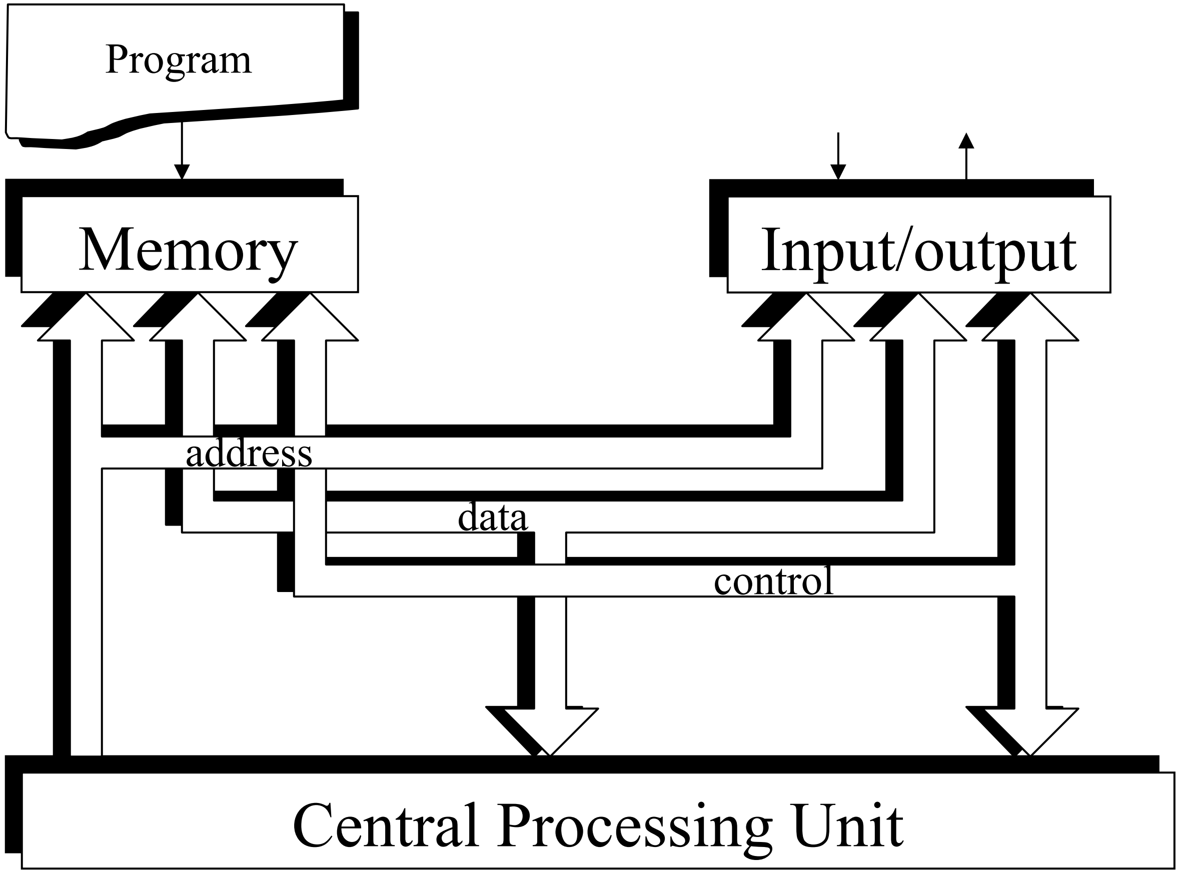
A software program to implement the proposed solution will be implemented in the PLC.

Architecture of PLCs



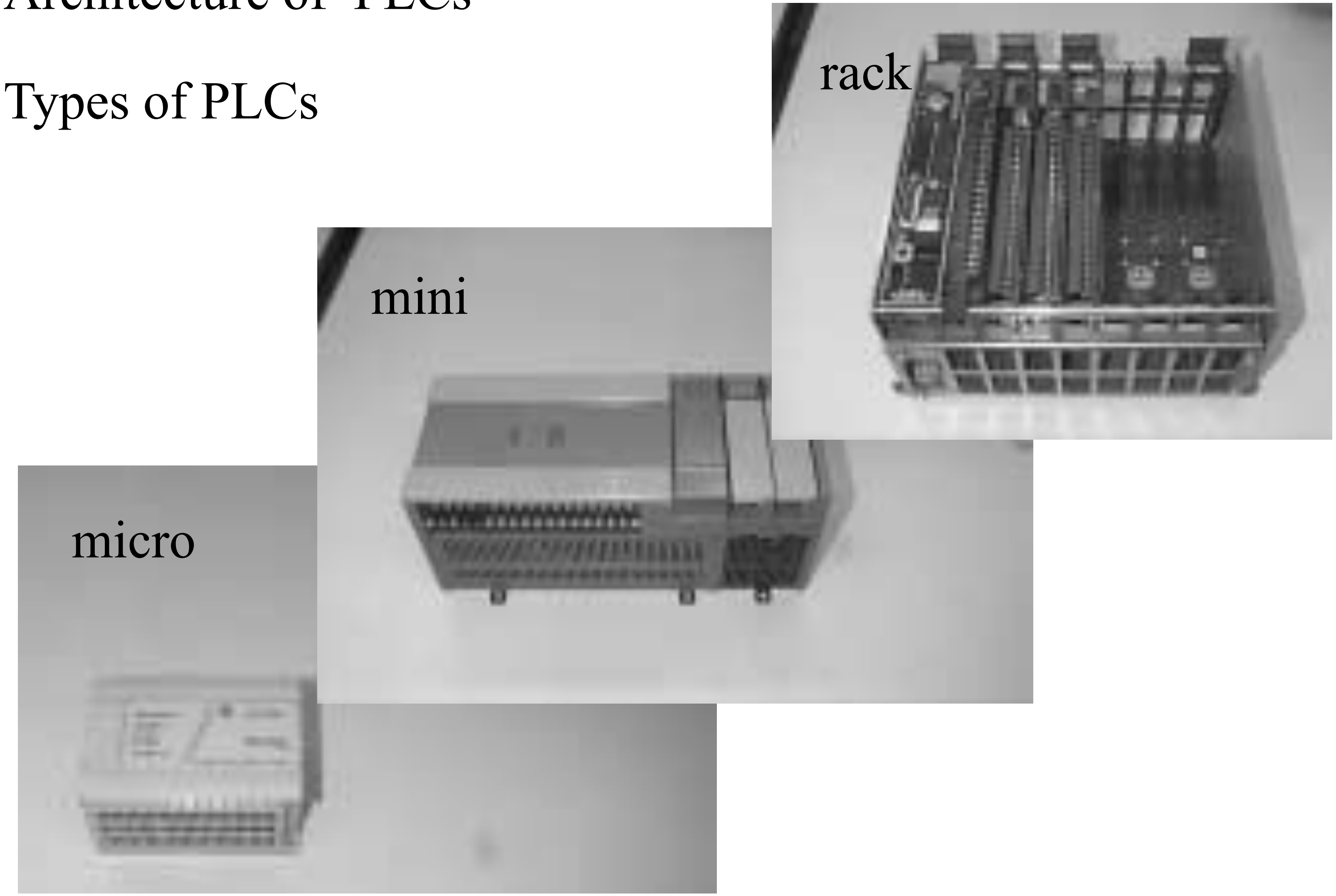
Architecture of PLCs

... and internally, how is it implemented?



Architecture of PLCs

Types of PLCs



Architecture of PLCs

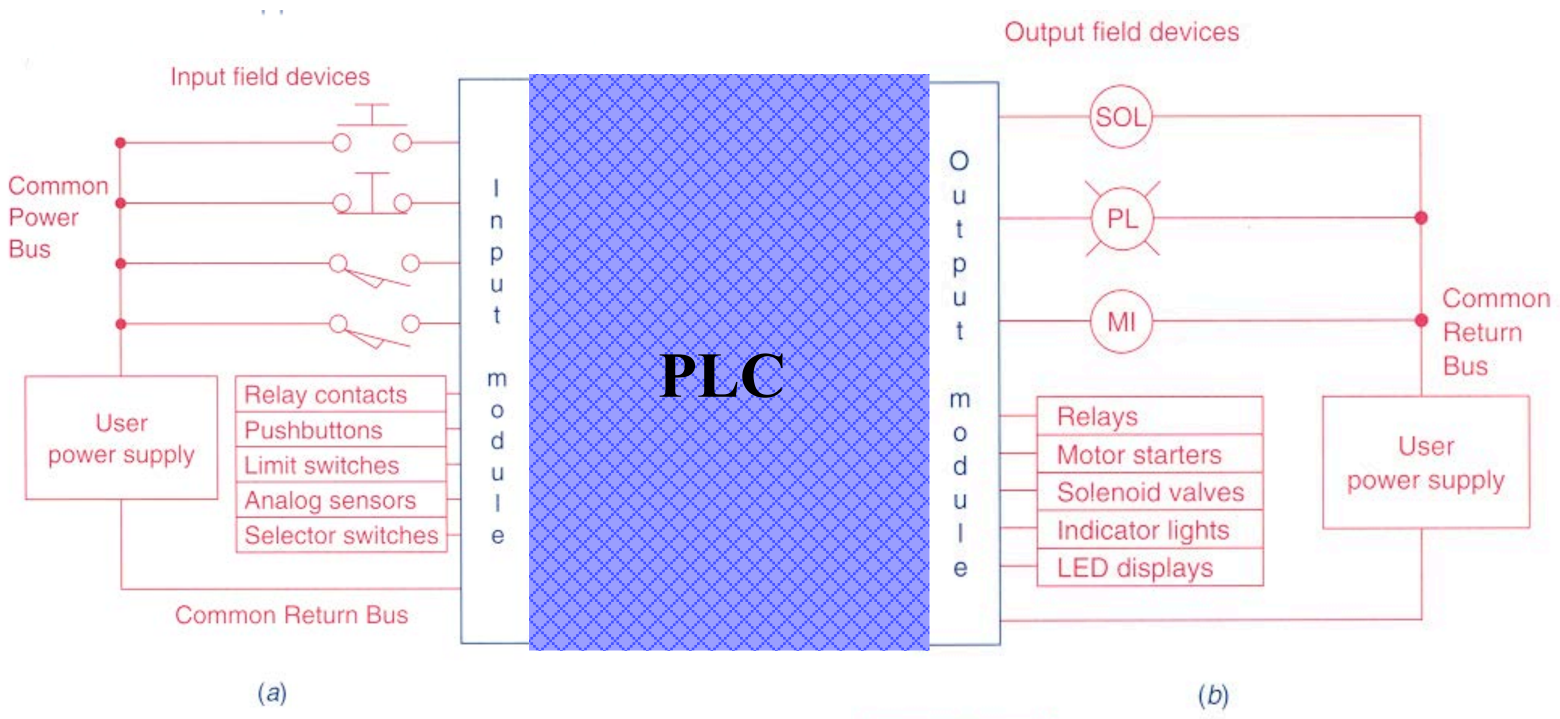
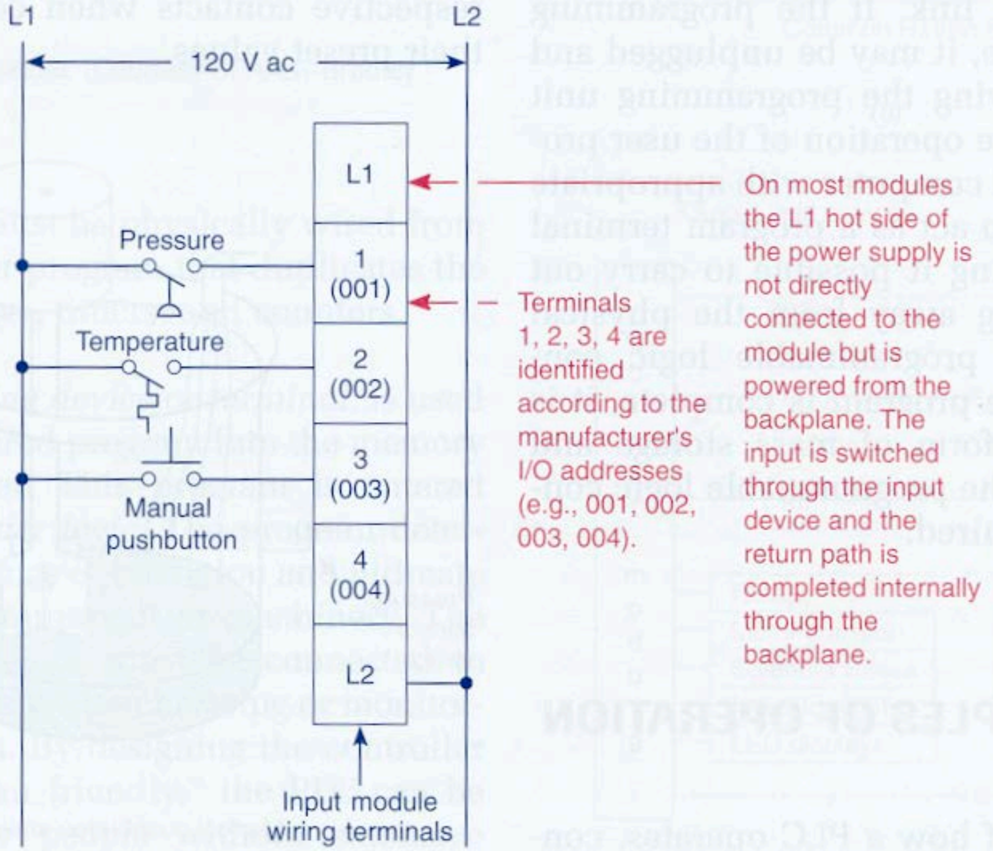


Fig. 1-6

(a) Typical input module. (b) Typical output module.

Architecture of PLCs



Terminals 1, 2, 3, 4 are identified according to the manufacturer's I/O addresses (e.g., 001, 002, 003, 004).

On most modules the L1 hot side of the power supply is not directly connected to the module but is powered from the backplane. The input is switched through the input device and the return path is completed internally through the backplane.

Fig. 1-10

Typical input module wiring connections.

Components of Programmable Logic Controllers

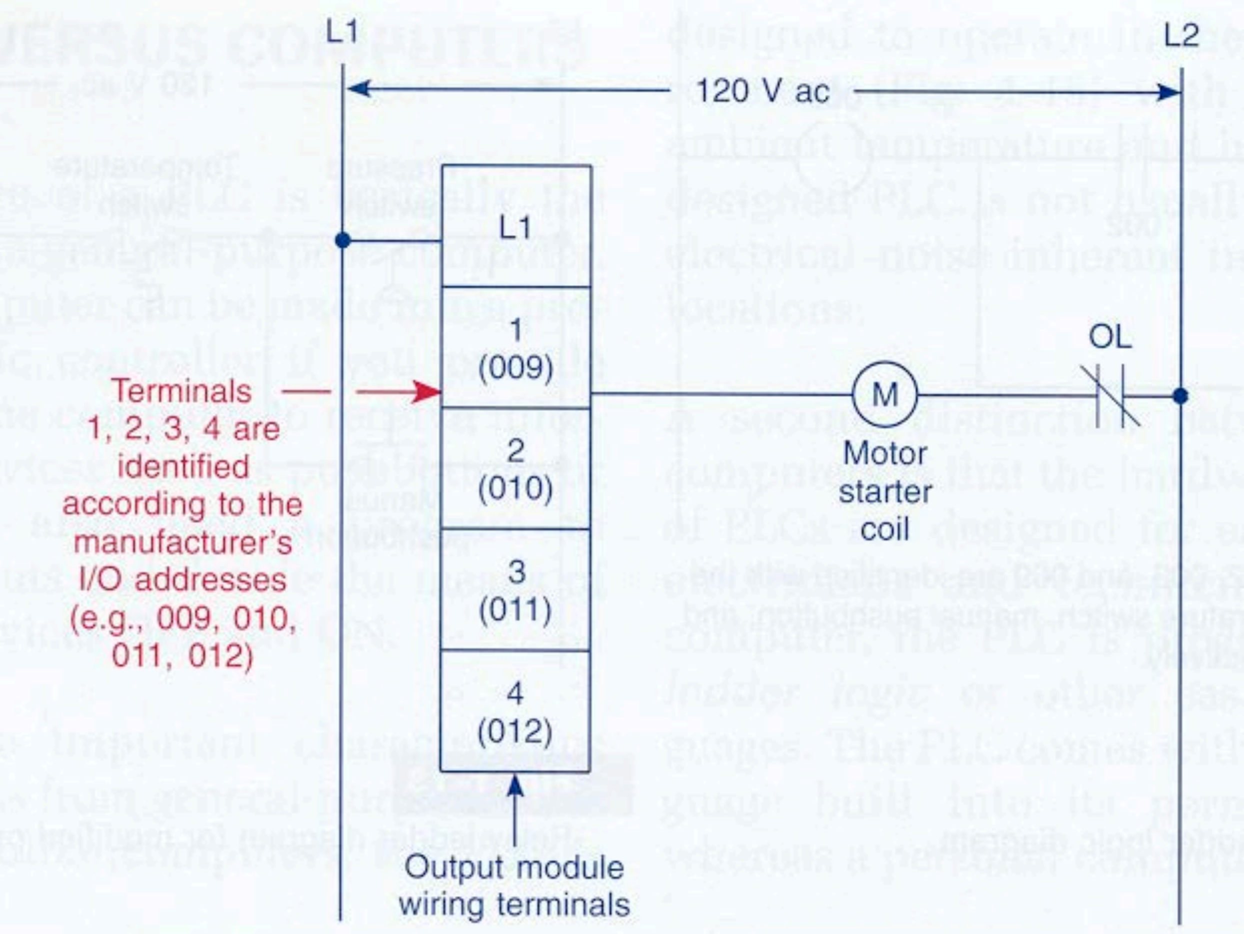

L1
L2
120 V ac
L1
1
(009)
2
(010)
3
(011)
4
(012)
Terminals 1, 2, 3, 4 are identified according to the manufacturer's I/O addresses (e.g., 009, 010, 011, 012)
M
Motor starter coil
OL
Output module wiring terminals

Fig. 1-11

Typical output module wiring connections.

Components of Programmable Logic Controllers

Example:

Command of a motor from a console with start and stop buttons.

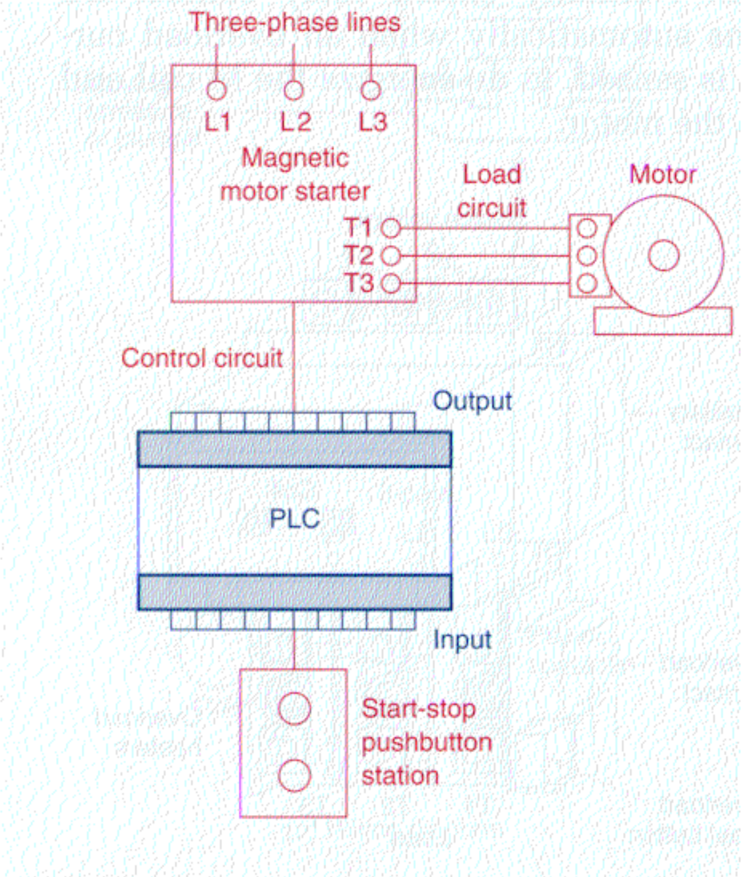


Fig. 6-9

When a PLC needs to control a large motor, it must work in conjunction with a starter.

Components of Programmable Logic Controllers

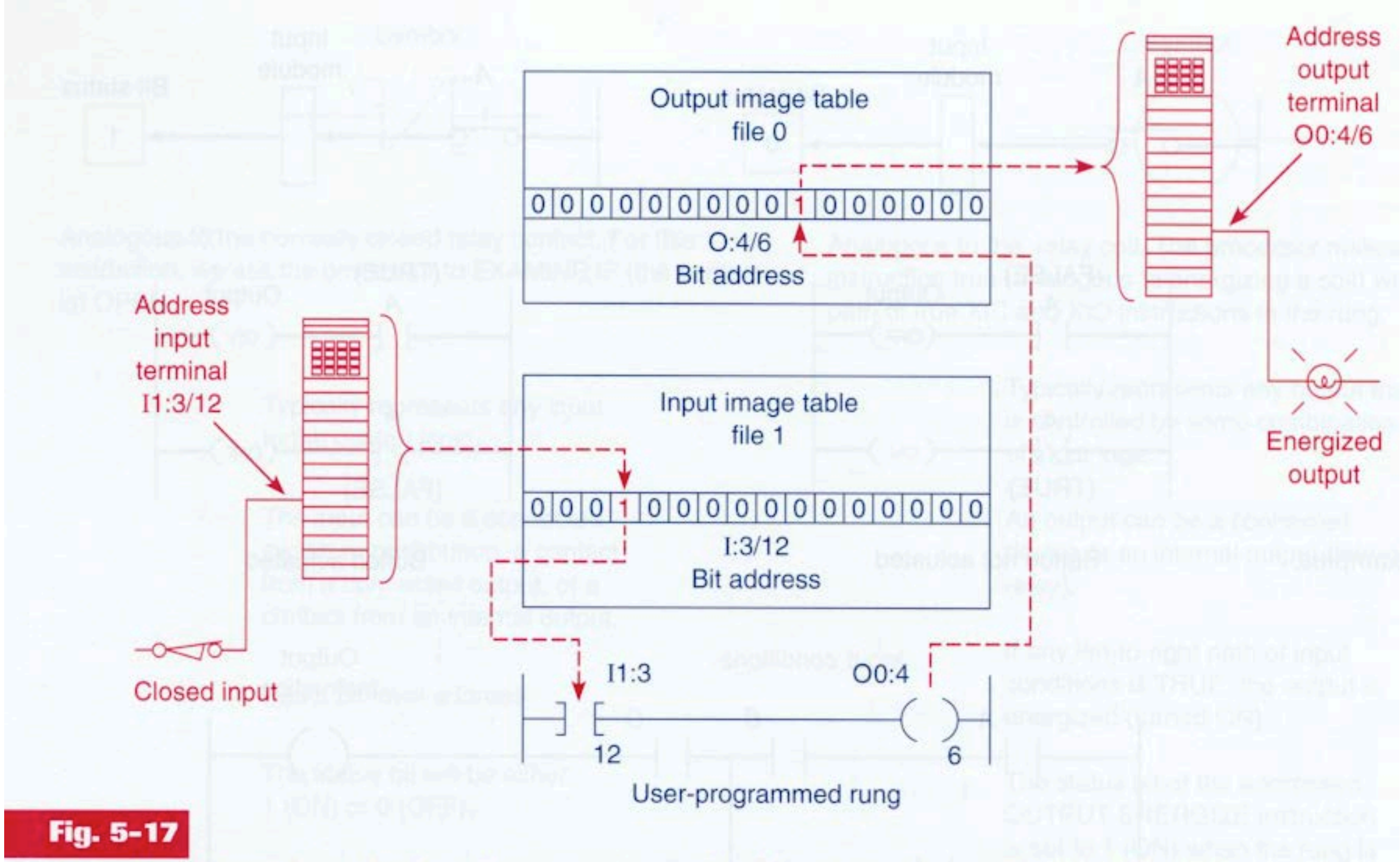
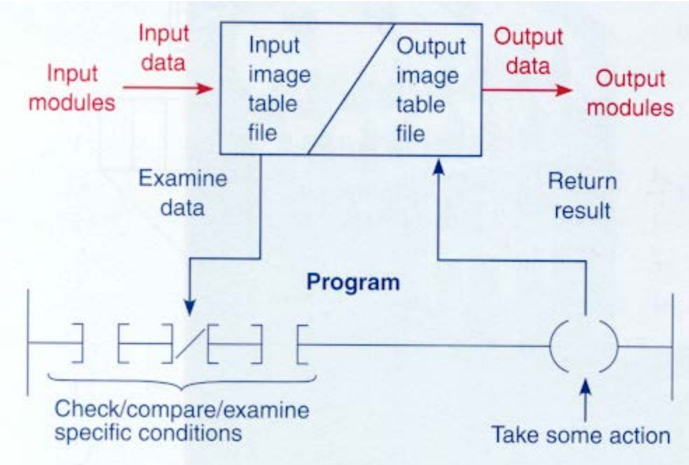


Fig. 5-17 The address identifies a location in the prossessor's data files, where the on/off state of the bit is stored.

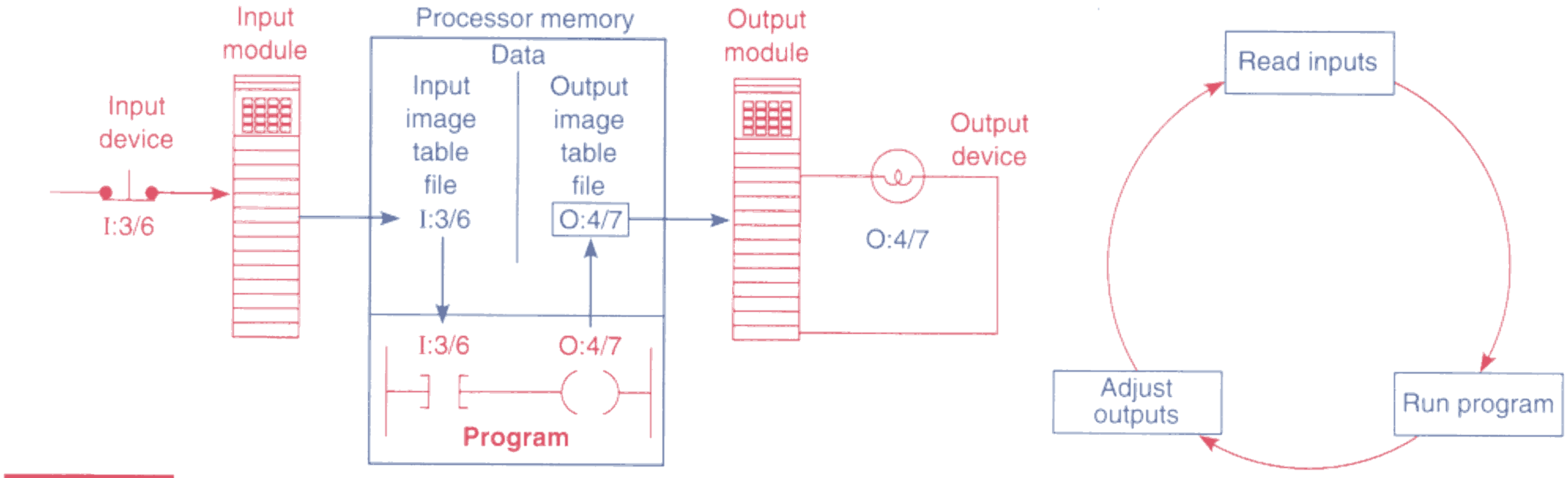
Internal structure

and

Work principles



(a) Data flow overview

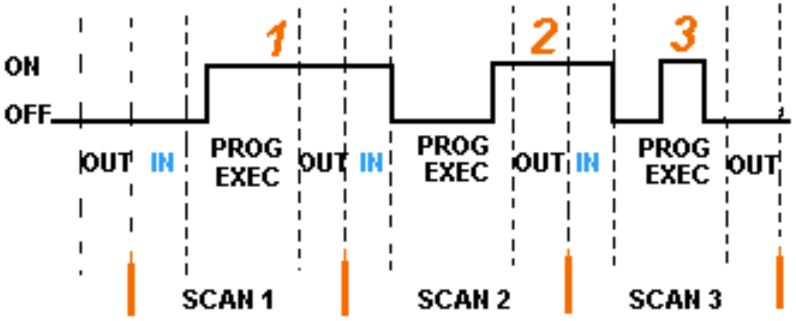


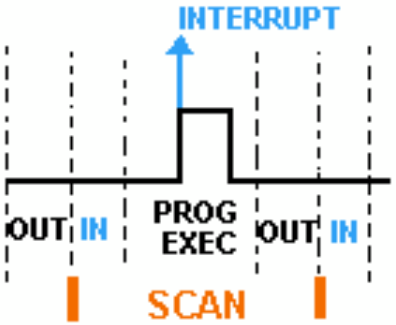
(b) Scan cycle

Fig. 5-7

Scan process.

Internal structure and work principles



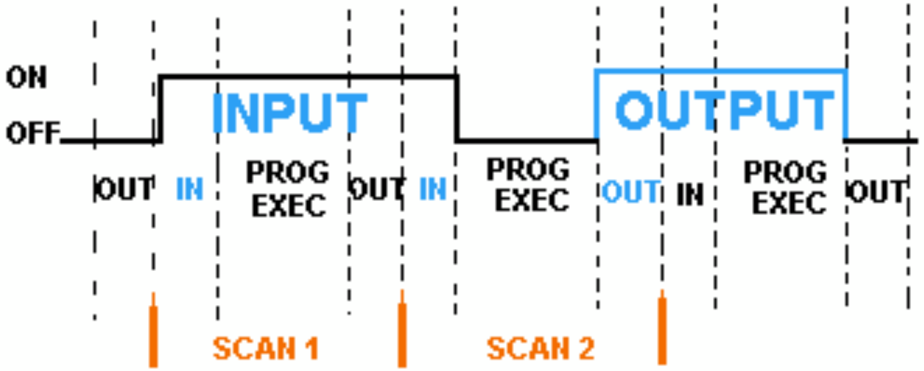


Scan Cycle, Scan Period

The inputs must be actives for at least one scan cycle to have impact (no uncertainty) in the internal PLC state and indirectly in the outputs.

Exception: interrupts...

Internal structure and work principles



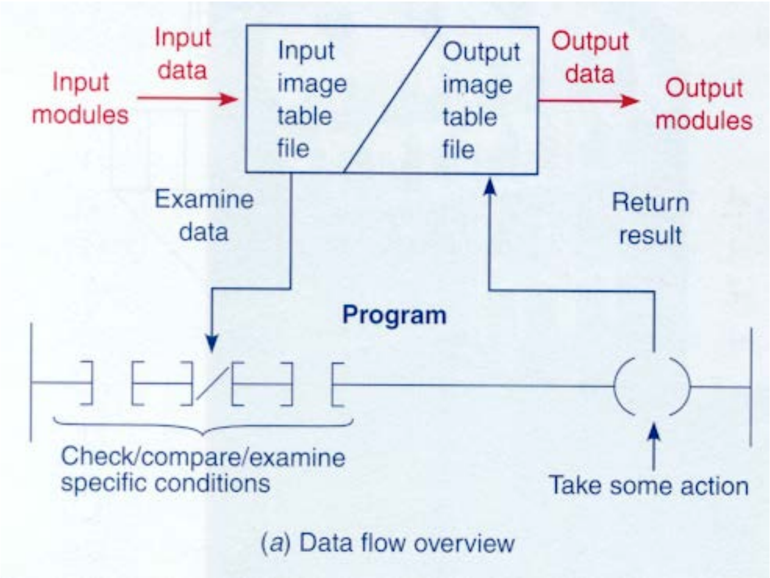
Time interval for an input to have inpact on an output (with probability one)?

2 * SCAN PERIOD

Smaller time interval (with probability greater than zero) that the change in one input can impact in one output?

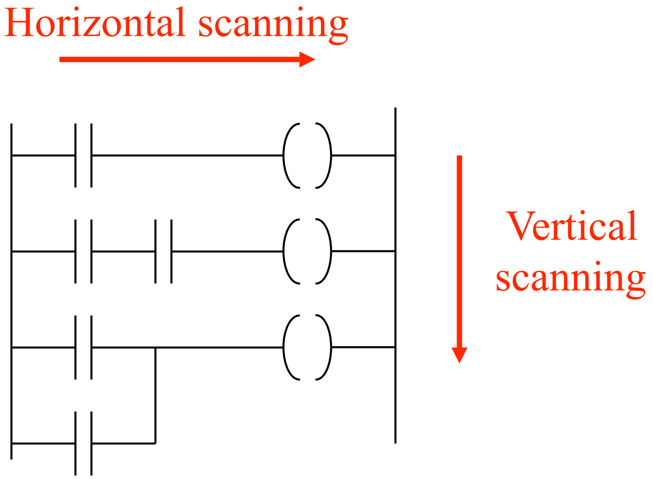
SCAN PERIOD – READ TIME – WRITE TIME = EXECUTION TIME

Internal structure and work principles



(*a*) Data flow overview

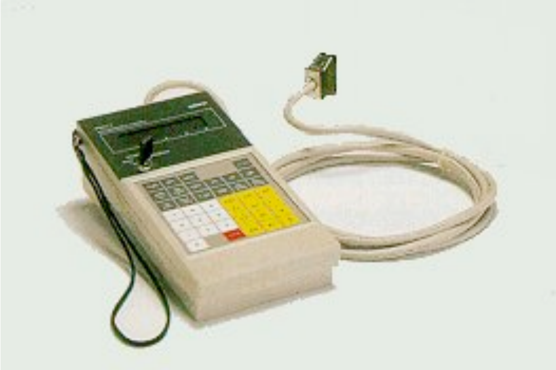
Interface for inputs and outputs



Scanning rangs...

Components of Programmable Logic Controllers

Programming using specific devices



OMRON console

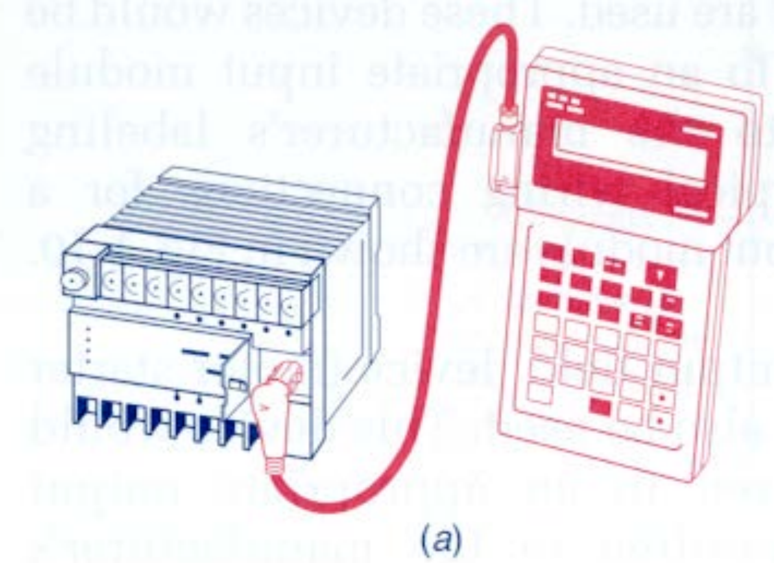
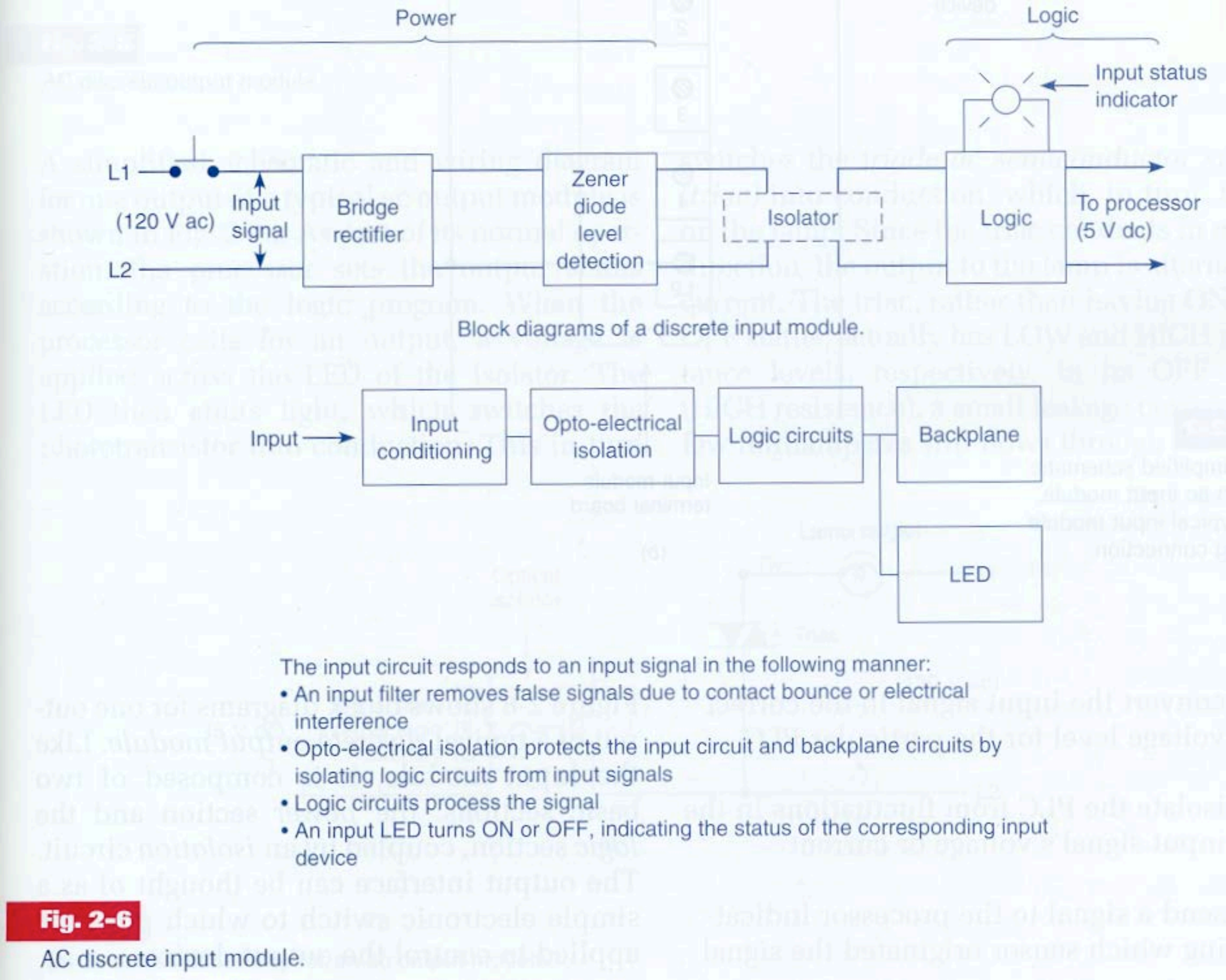


Fig. 1-7

Programming devices: (a) hand-held unit with light-emitting diode (LED) display; (b) industrial terminal video unit (Courtesy of Honeywell, Inc.); (c) personal computer with appropriate software.

Input and output interfaces

AC input module (discrete)



Block diagrams of a discrete input module.

The input circuit responds to an input signal in the following manner:

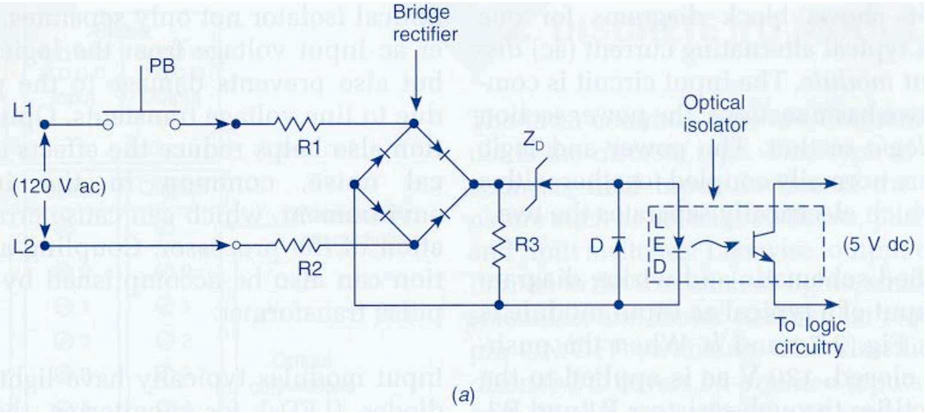
- An input filter removes false signals due to contact bounce or electrical interference
- Opto-electrical isolation protects the input circuit and backplane circuits by isolating logic circuits from input signals
- Logic circuits process the signal
- An input LED turns ON or OFF, indicating the status of the corresponding input device

Fig. 2-6

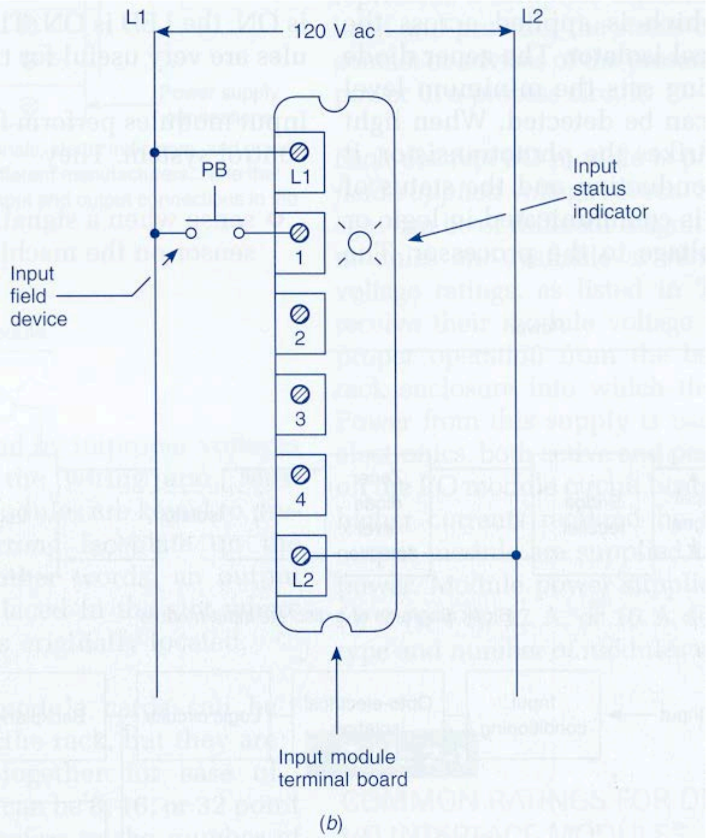
AC discrete input module.

Input and output interfaces

AC input module:
simplified implementation



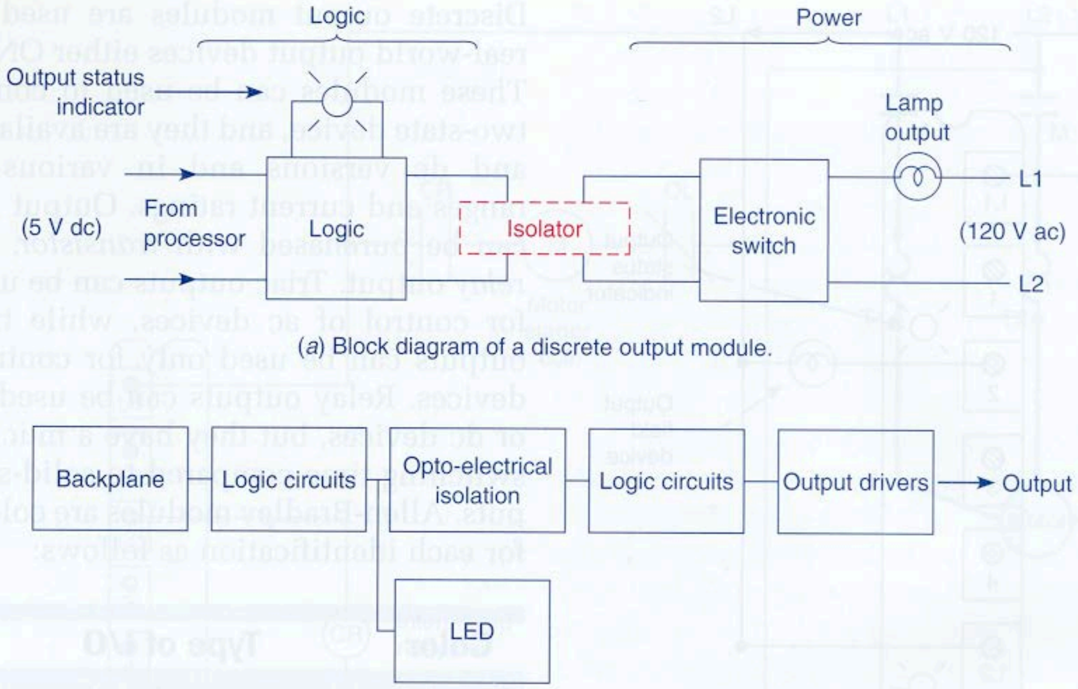
Electronic circuit



Connections to the PLC terminals

Input and output interfaces

AC output module (discrete)



(*a*) Block diagram of a discrete output module.

(*b*) The output circuit controls the output signal in the following manner:

- Logic circuits determine the output status
- An output LED indicates the status of the output signal
- Opto-electrical isolation separates output circuit logic and backplane circuits from field signals
- The output driver turns the corresponding output ON or OFF.

Fig. 2-8

AC discrete output module.

Input and output interfaces

AC output module (discrete)

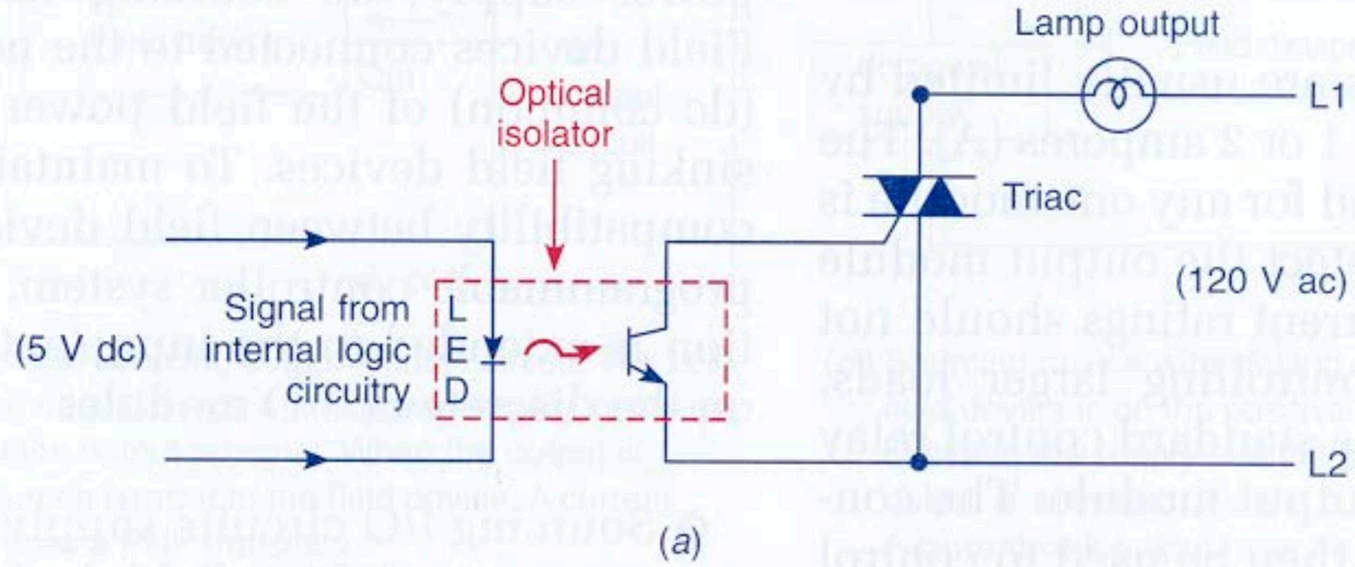


Fig. 2-9

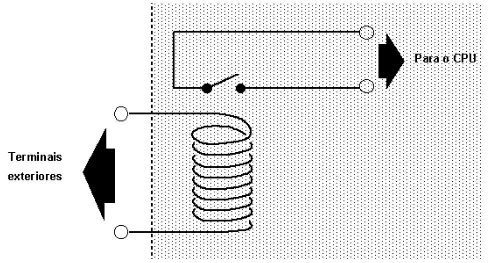
(a) Simplified schematic for an ac output module.

Circuito electrónico (simplificado)

Input and output interfaces

DC input module (discrete)

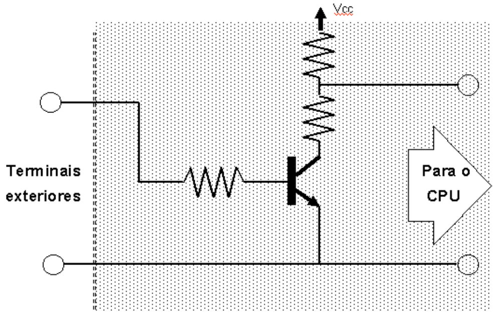
Relay



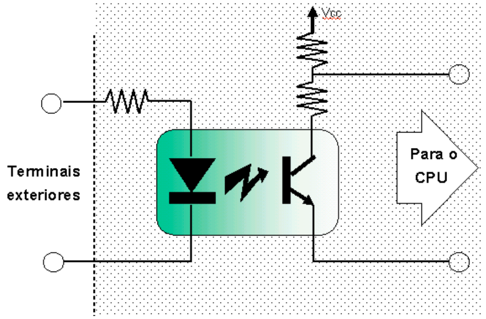
Attention to:

- Galvanic isolation
- Economy
- Consumption
- Switching speed
- Noise imunity

Transistor



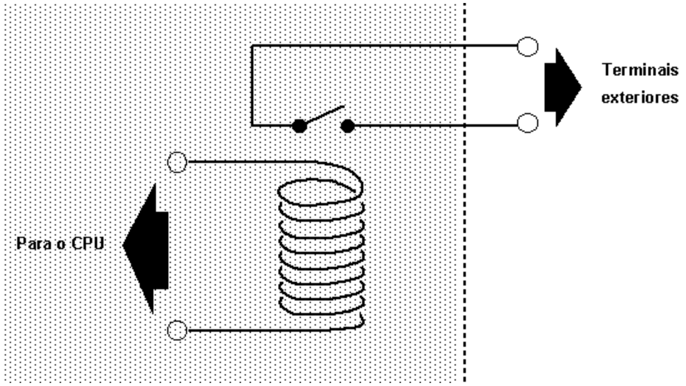
Opto-coupler



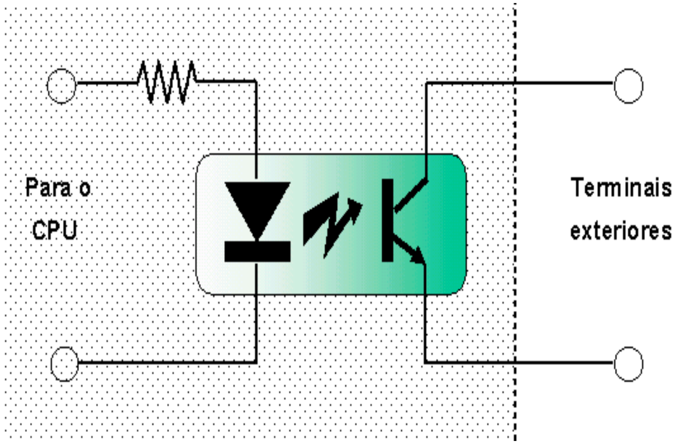
Input and output interfaces

DC output module (discrete)

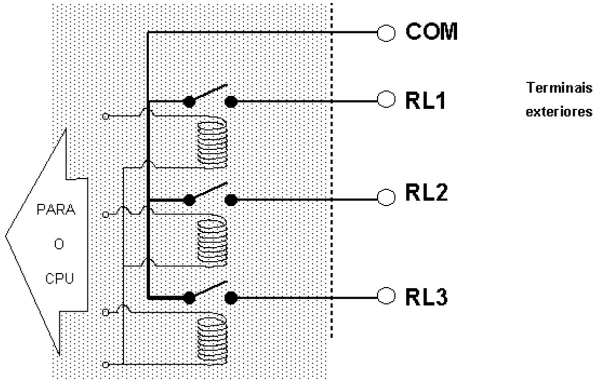
Relay



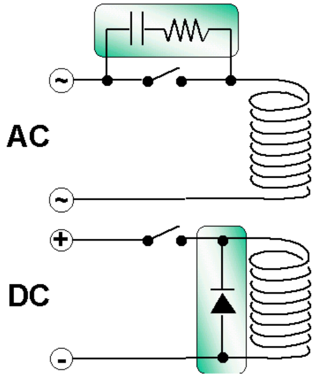
Transistor



Connections to terminals ...



... and protections.

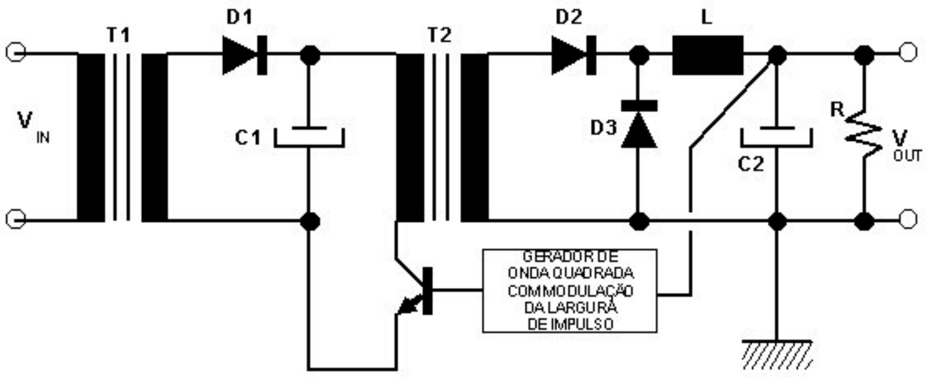


Components of Programmable Logic Controllers

Power sources

Attention to:

- Isolation to the noise
- Isolation relative to disturbances on the network
- Efficiency
- Consumption
- Size (volume and weight)
- Robustness relative to load variations


T1
D1
T2
D2
L
V_{IN}
C1
D3
R
V_{OUT}
C2
GERADOR DE ONDA QUADRADA COM MODULAÇÃO DA LARGURA DE IMPULSO

Switching power sources

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.*

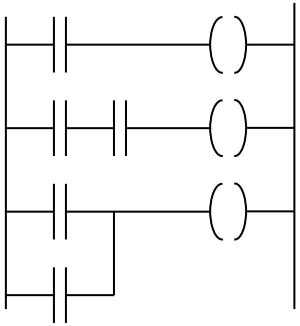
Software development resources.

...

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

PLCs Programming Languages (IEC 1131-3)

Ladder Diagram



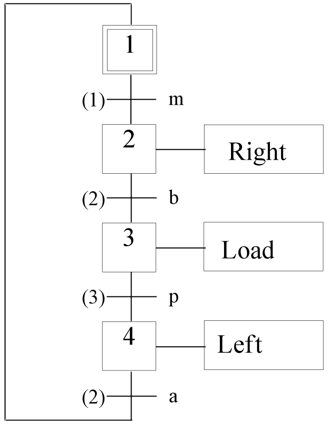
Structured Text

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

Instruction List

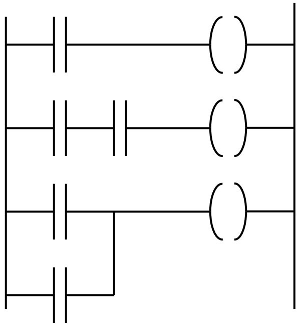
```
LD        %M12
AND       %I1.0
ANDN      %I1.1
OR        %M10
ST        %Q2.0
```

Sequential Function Chart (GRAFCET)



Linguagens de programação de PLCs (IEC 1131-3)

Ladder Diagram



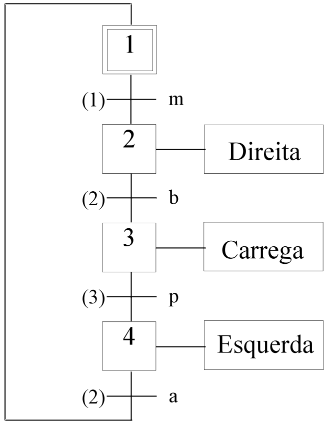
Structured Text

```
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

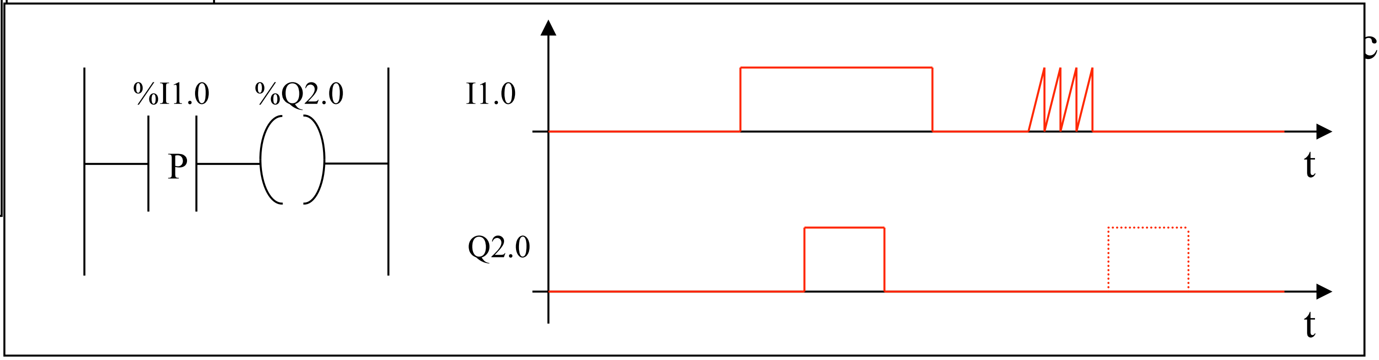
```
ANI1
A(
OI2
O(
ANC9
AQ9
)
)
=Q9
...
AI3
=P9
NO
OM1
OI4
=Z9
NO
AC9
=M1
...
LDV50
=CSW9
PE
```

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

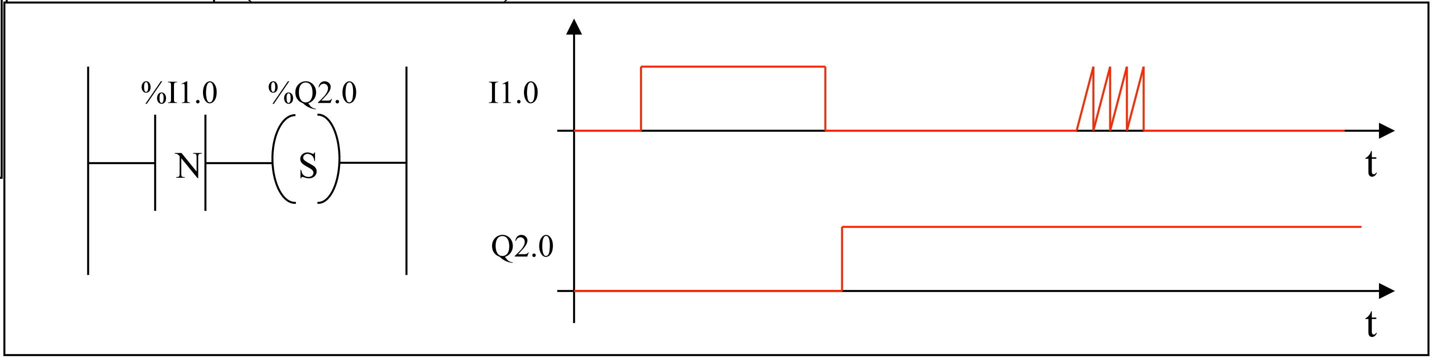


Instruction list

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		



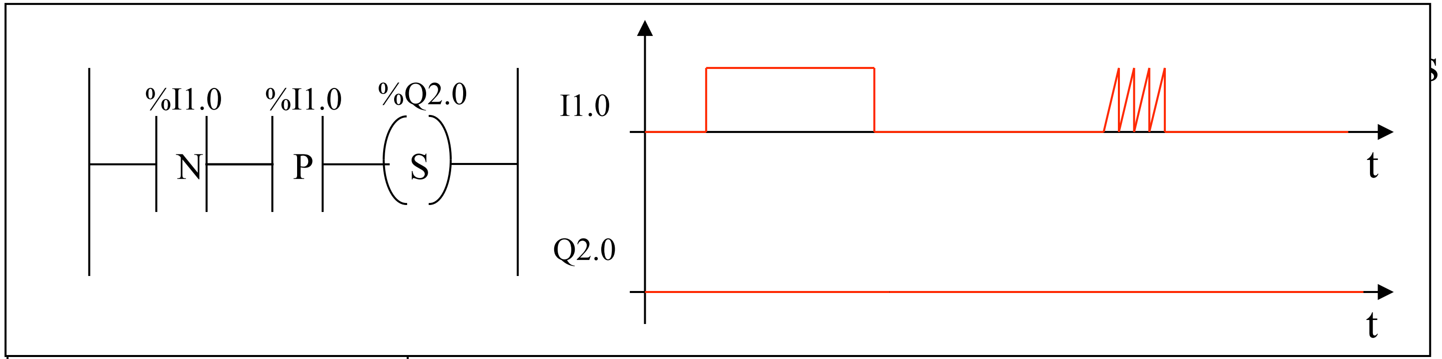
Instruction list

Basic Instructions

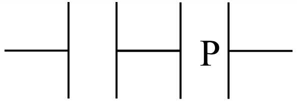
AND

AND

ANDN

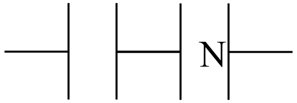


ANDR



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

ANDF

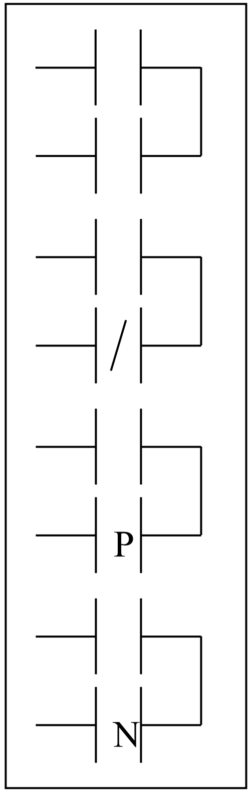


AND of the falling edge with the result of the previous logical operation.

Instruction list

Basic Instructions

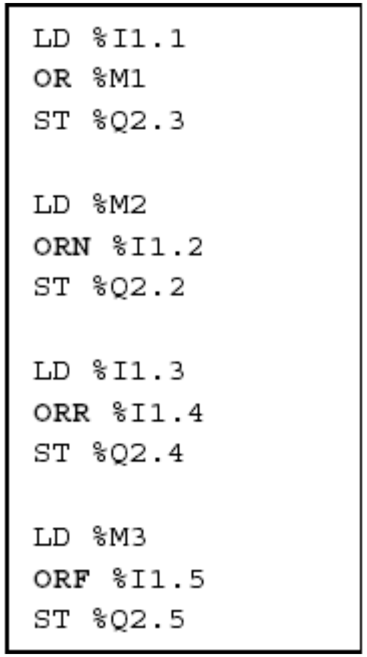
OR



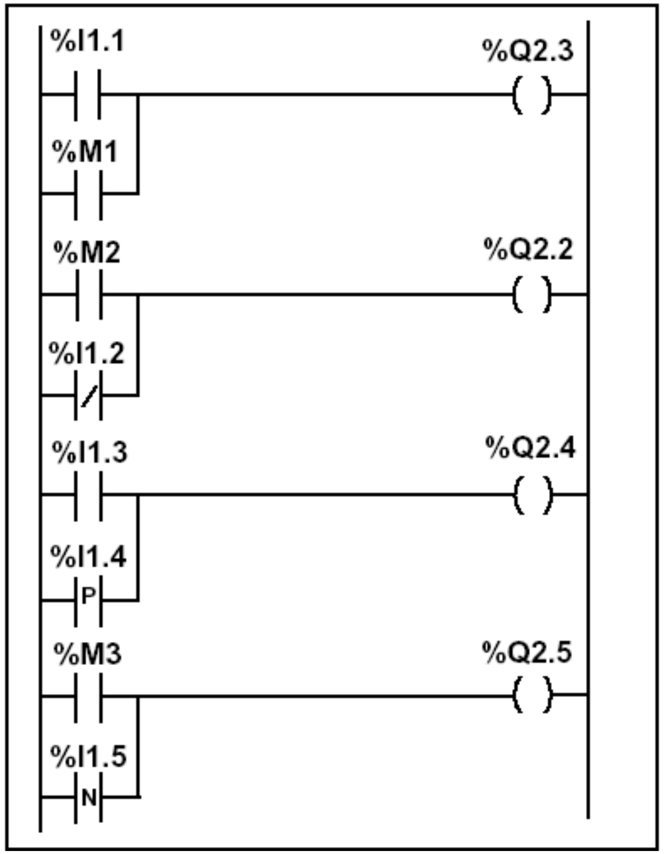
OR	OR of the operand with the result of the previous logical operation.
ORN	OR of the operand with the inverted result of the previous logical operation.
ORR	OR of the rising edge with the result of the previous logical operation.
ORF	OR of the falling edge with the result of the previous logical operation.

Instruction list

Example:



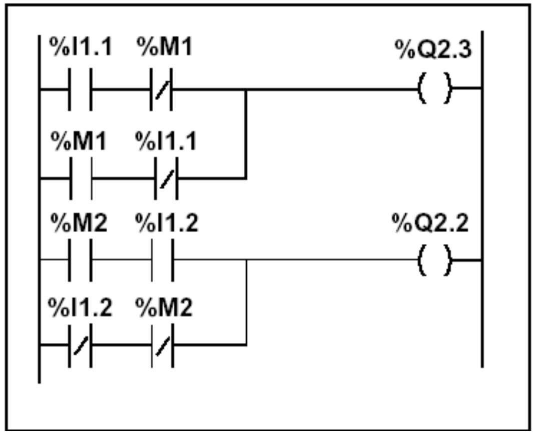
```
LD %I1.1
OR %M1
ST %Q2.3

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
```



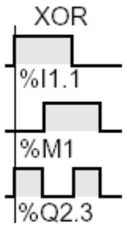
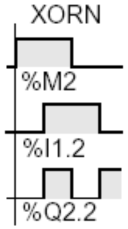
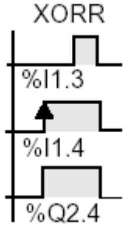
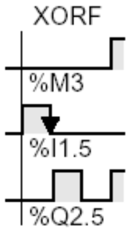
Instruction list

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

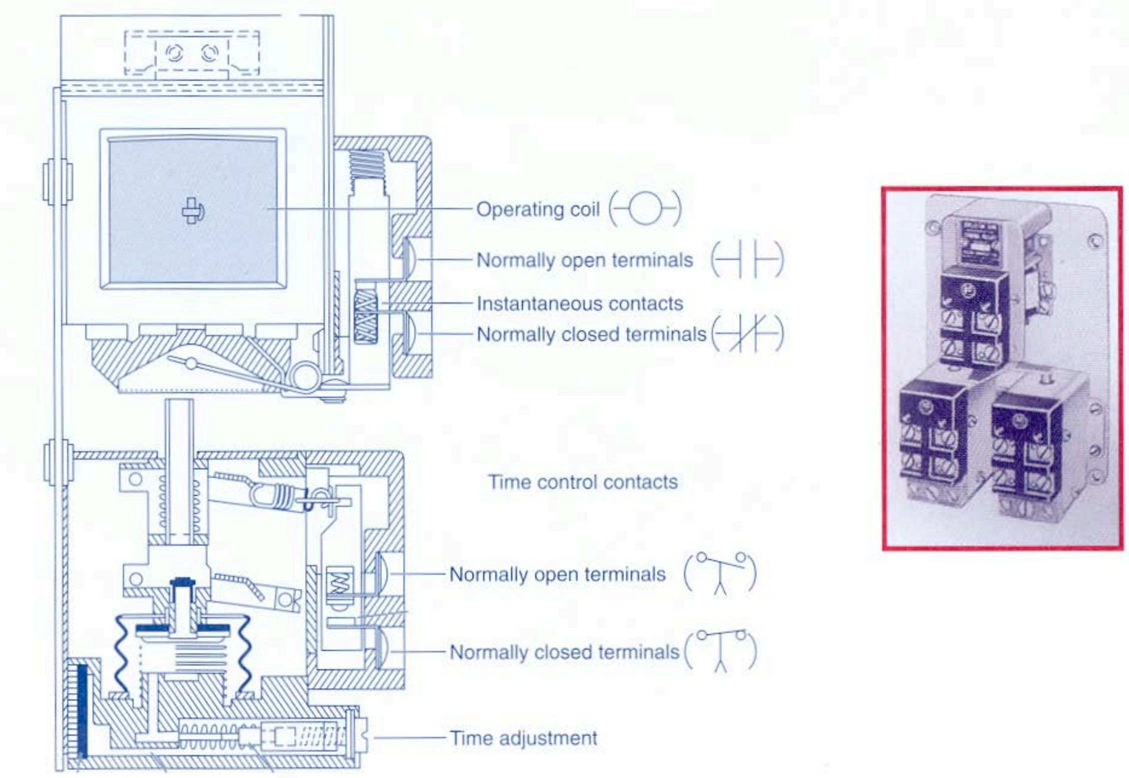


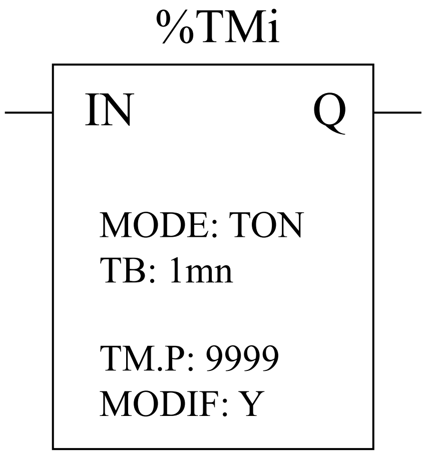
Fig. 7-1
Pneumatic on-delay timer. *(Courtesy of Allen-Bradley Company, Inc.)*

Instruction list

Temporized Relays

or

Timers


%TMi

IN Q

MODE: TON
TB: 1mn

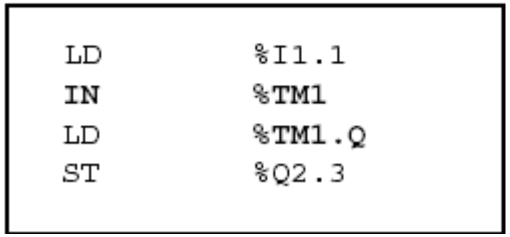
TM.P: 9999
MODIF: Y

Characteristics:

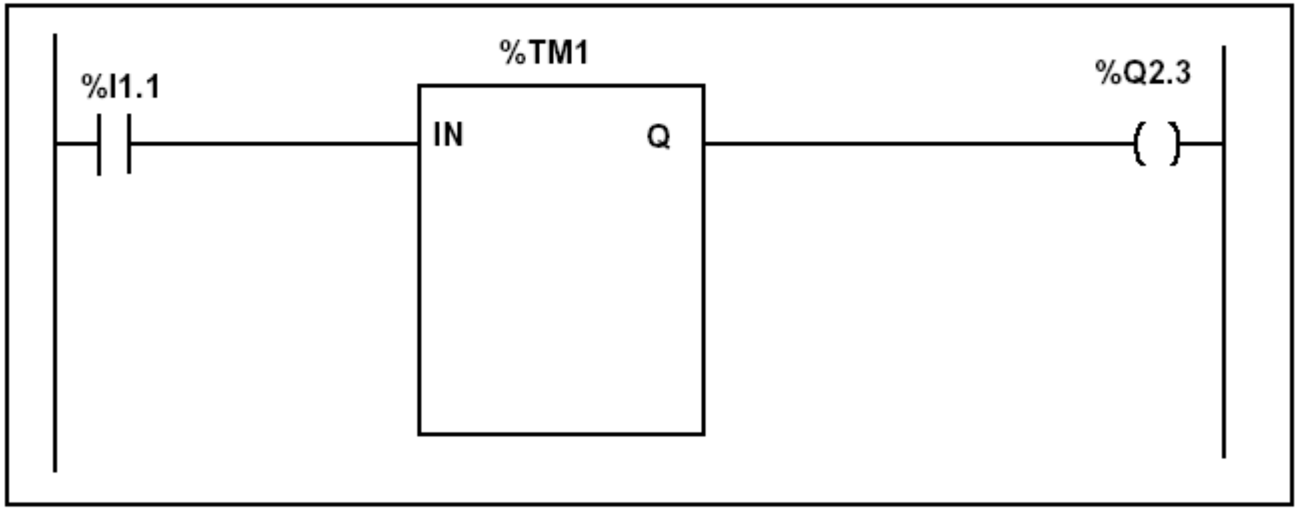
Identifier:%TMi	0..63 in the TSX37	
Input:	IN	to activate
Mode:	TON TOFF TP	On delay Off delay Monostable
Time basis:	TB	1mn (def.), 1s, 100ms, 10ms
Programmed value:	%TMi.P	0...9999 (def.) period=TB*TMi.P
Actual value:	%TMi.V	0...TMi.P (can be real or tested)
Modifiable:	Y/N	can be modified from the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

Instruction list

Relés temporizados
Ou
Timers

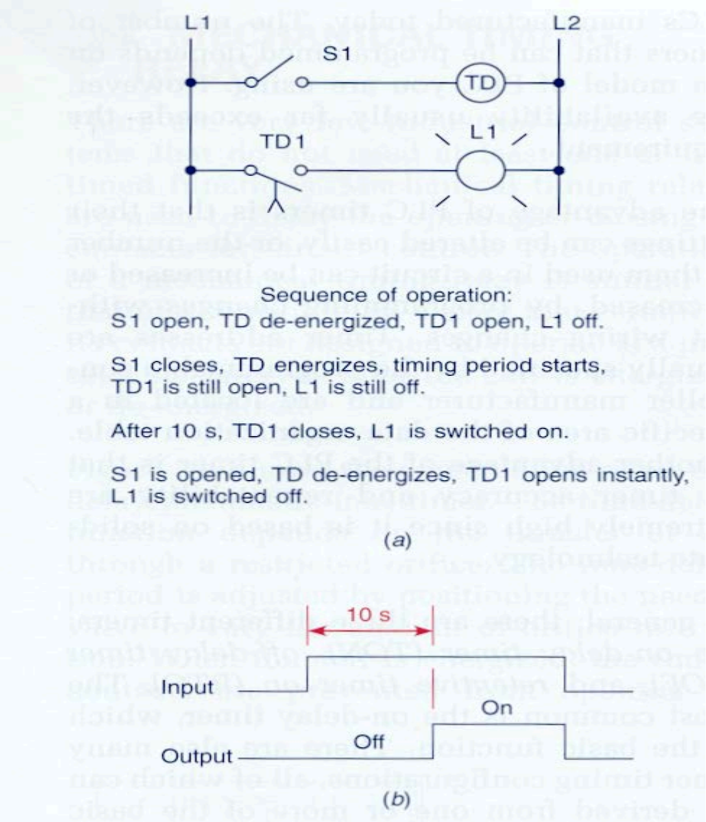


```
LD          %I1.1
IN          %TM1
LD          %TM1.Q
ST          %Q2.3
```



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.

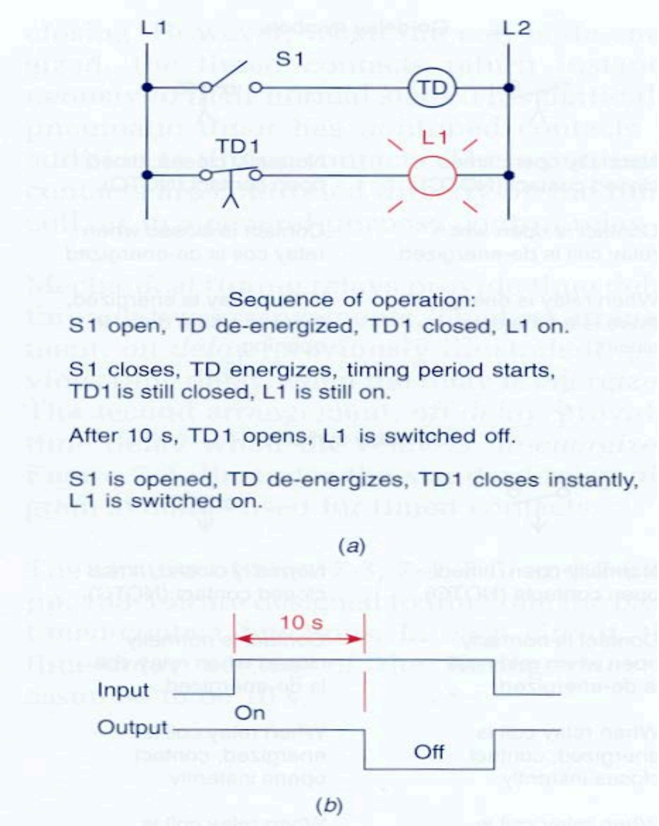
After 10 s, TD1 closes, L1 is switched on.

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

(a)

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.

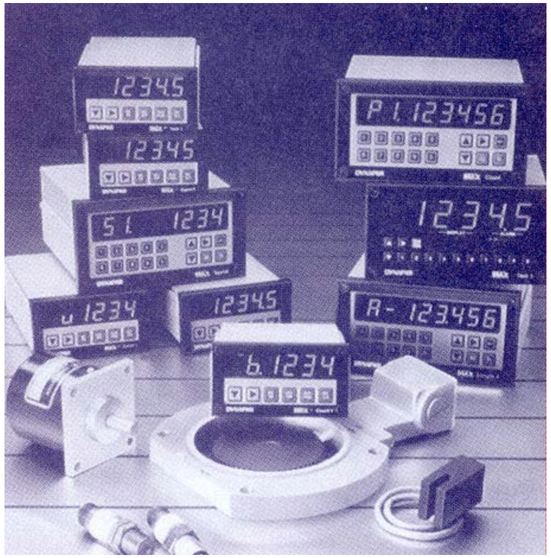
(a)

Fig. 7-4

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

Instruction list

Counters



Some applications...

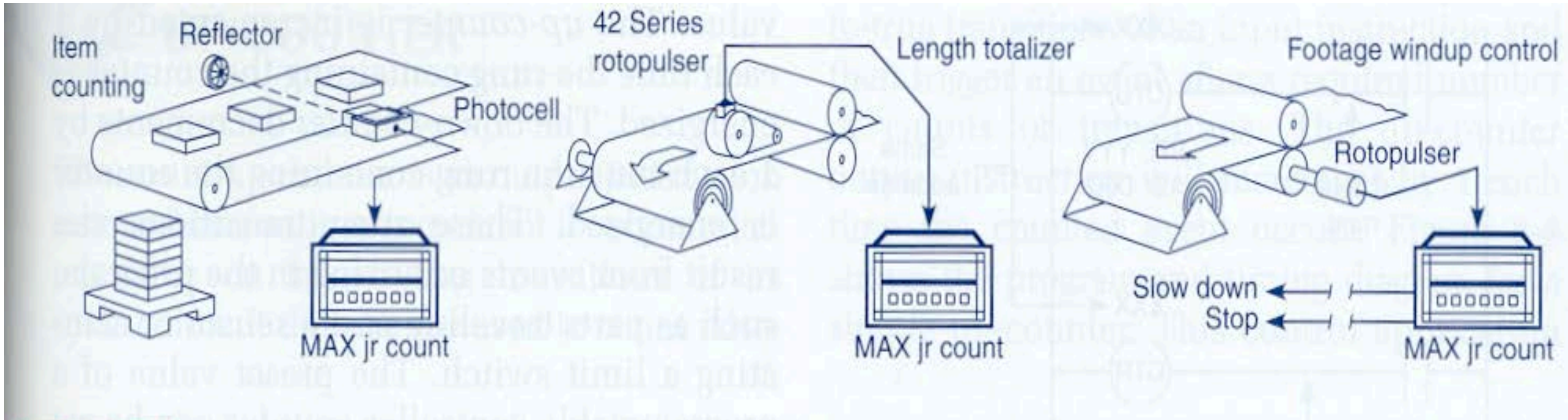
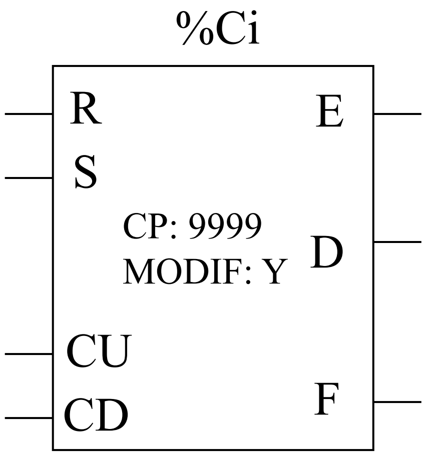


Fig. 8-3

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

Instruction list

Counters



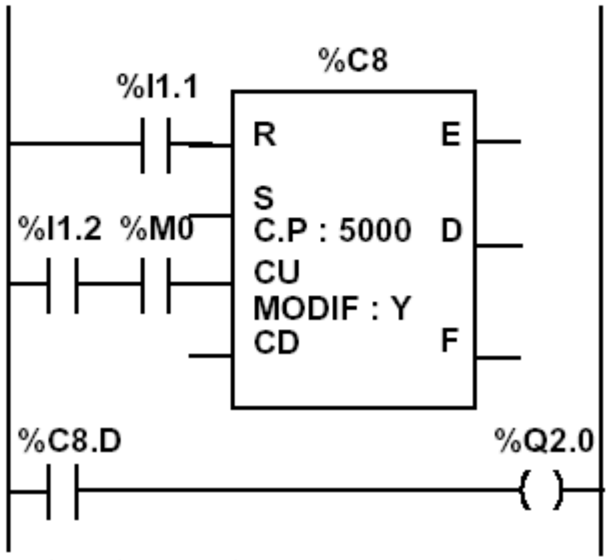
Characteristics:

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:



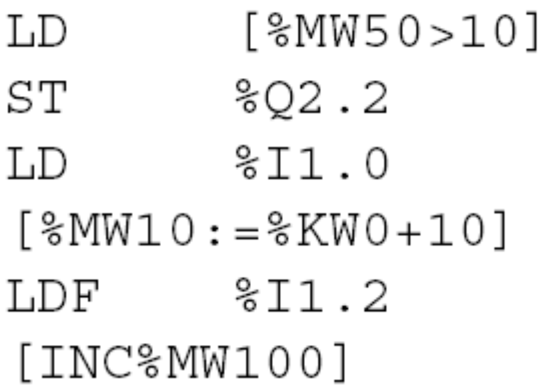
Instruction list language

```
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

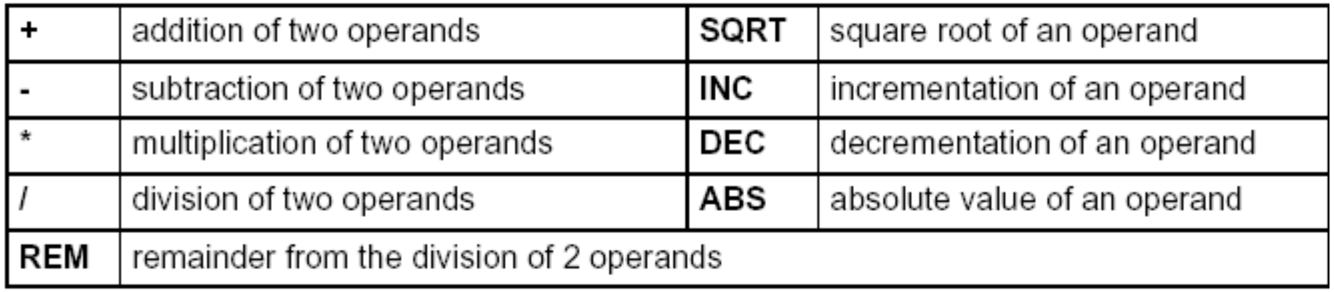


```
LD      [%MW50>10]
ST      %Q2.2
LD      %I1.0
[%MW10:=%KW0+10]
LDF     %I1.2
[INC%MW100]
```

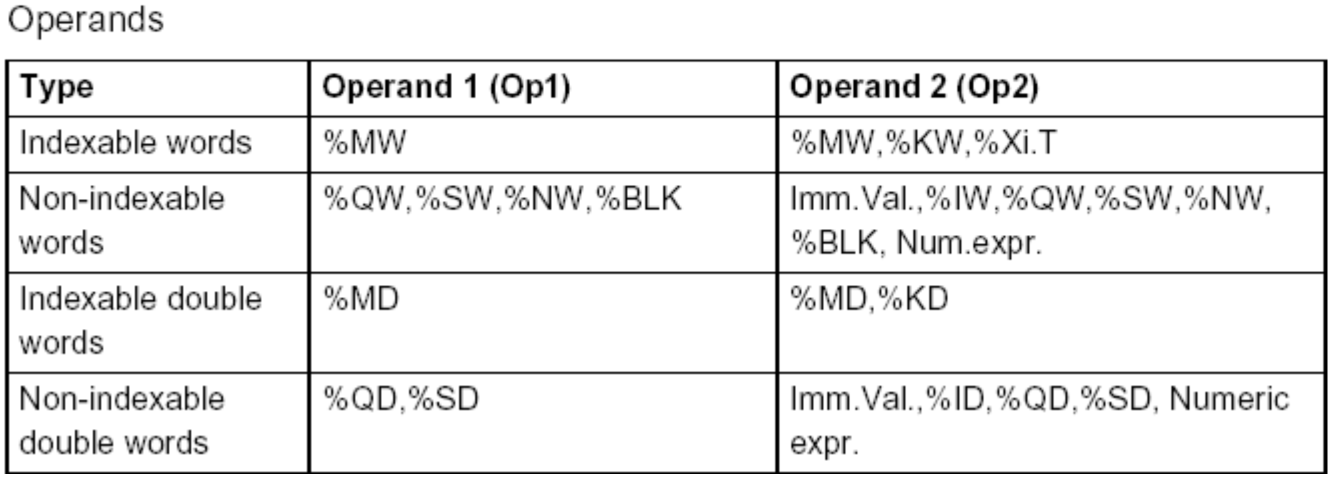
Instruction list

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

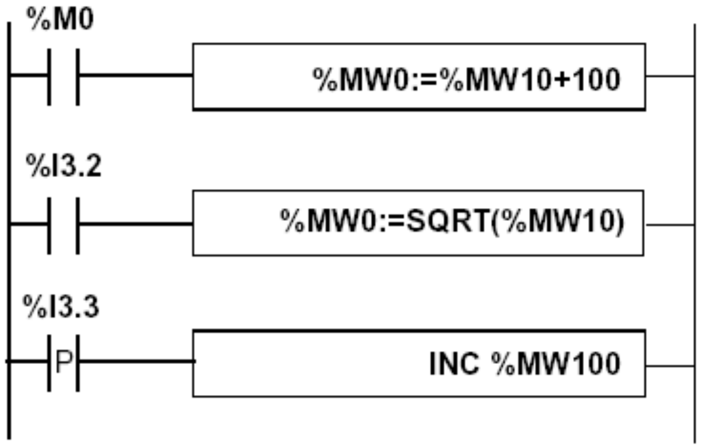
Type	**Operand 1 (Op1)**	**Operand 2 (Op2)**
Indexable words	%MW	%MW,%KW,%Xi.T
Non-indexable words	%QW,%SW,%NW,%BLK	Imm.Val.,%IW,%QW,%SW,%NW, %BLK, Num.expr.
Indexable double words	%MD	%MD,%KD
Non-indexable double words	%QD,%SD	Imm.Val.,%ID,%QD,%SD, Numeric expr.

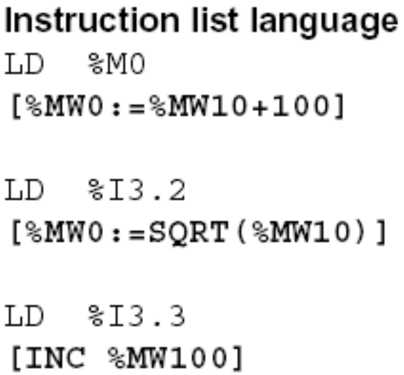
Instruction list

Numerical Processing

Example:

Arithmetic functions




Instruction list language

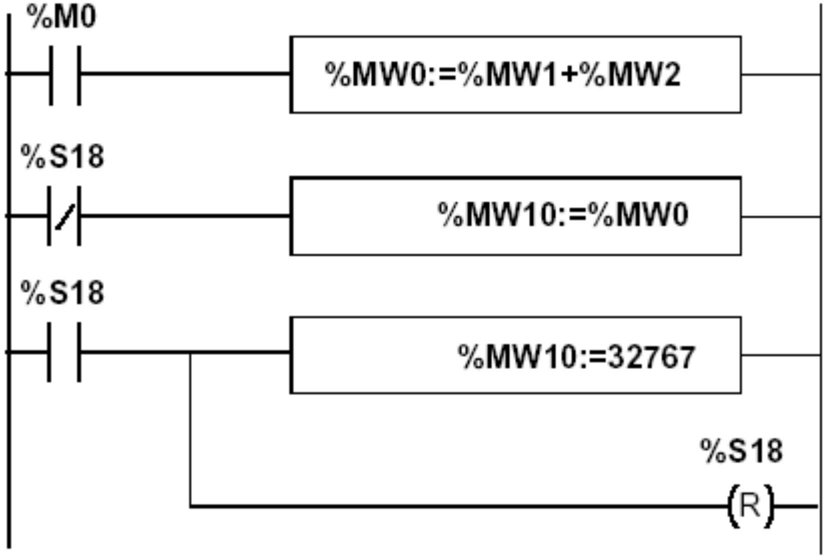
```
LD  %M0
[%MW0:=%MW10+100]

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

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

Instruction list

Numerical Processing

Example:

Arithmetic functions



Use of a system variable:

%S18 – flag de overflow

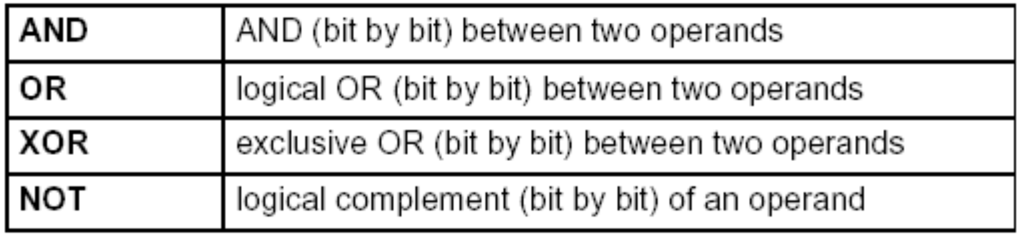
Example in instruction list language:

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

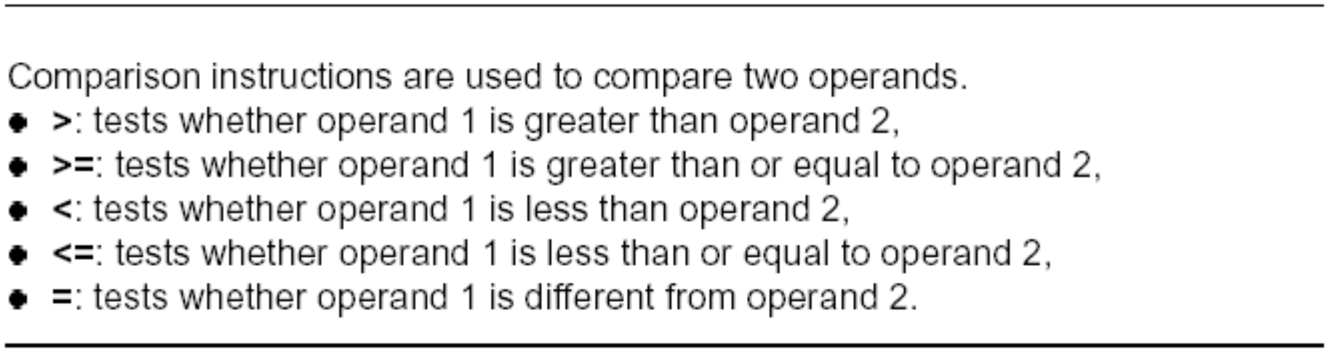
Instruction list

Numerical Processing

Logic Functions

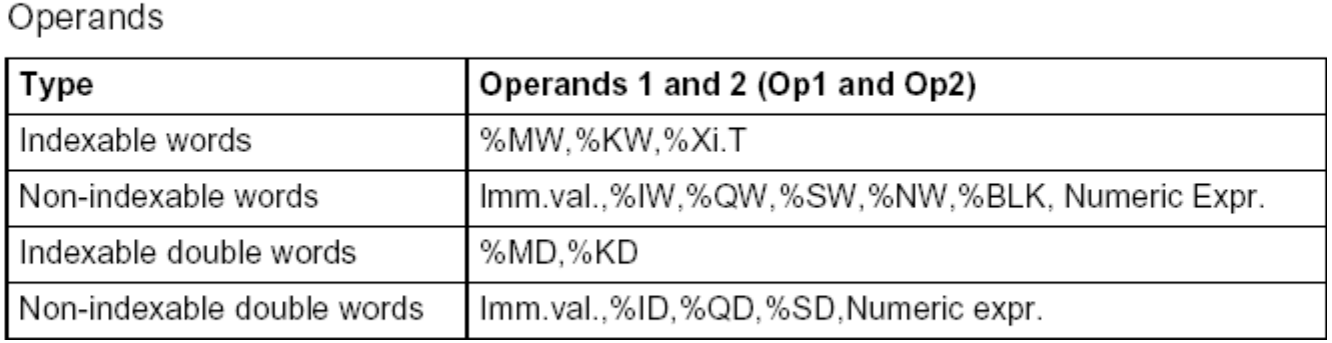


AND	AND (bit by bit) between two operands
OR	logical OR (bit by bit) between two operands
XOR	exclusive OR (bit by bit) between two operands
NOT	logical complement (bit by bit) of an operand



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

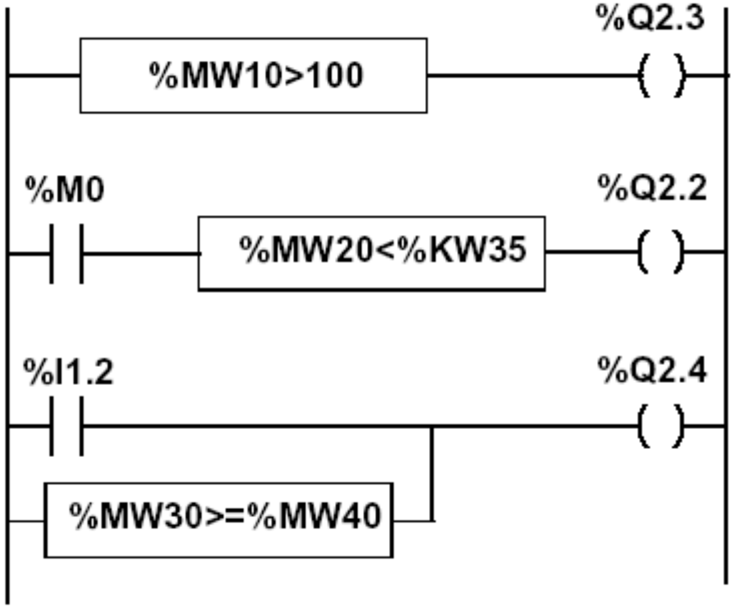
Type	Operands 1 and 2 (Op1 and Op2)
Indexable words	%MW,%KW,%Xi.T
Non-indexable words	Imm.val.,%IW,%QW,%SW,%NW,%BLK, Numeric Expr.
Indexable double words	%MD,%KD
Non-indexable double words	Imm.val.,%ID,%QD,%SD,Numeric expr.

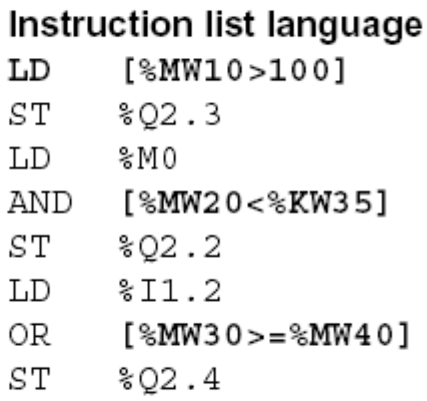
Instruction list

Numerical Processing

Example:

Logic functions



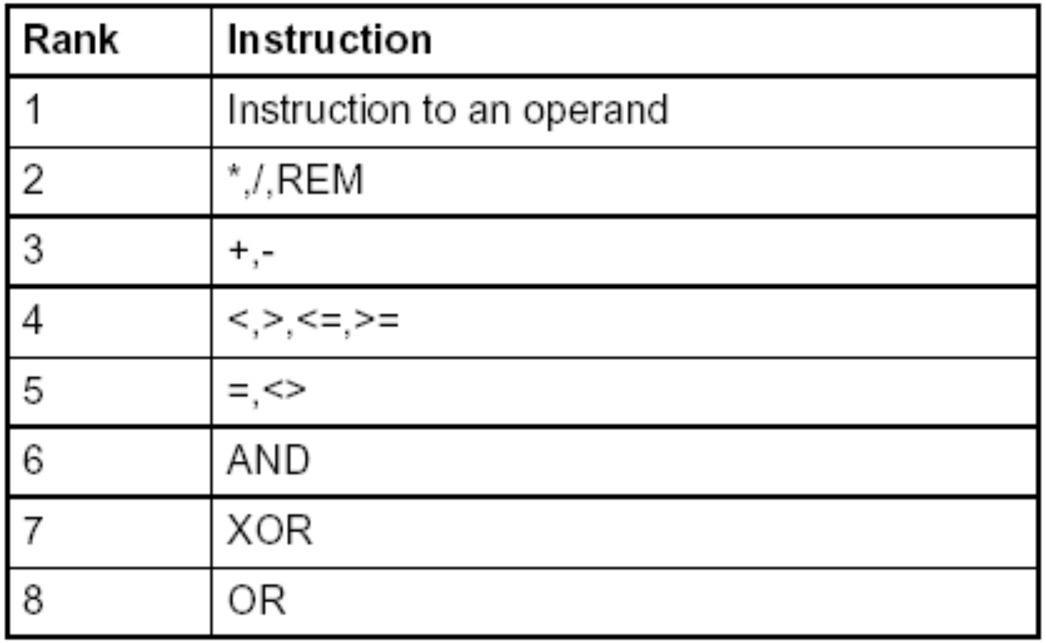

Instruction list language

```
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



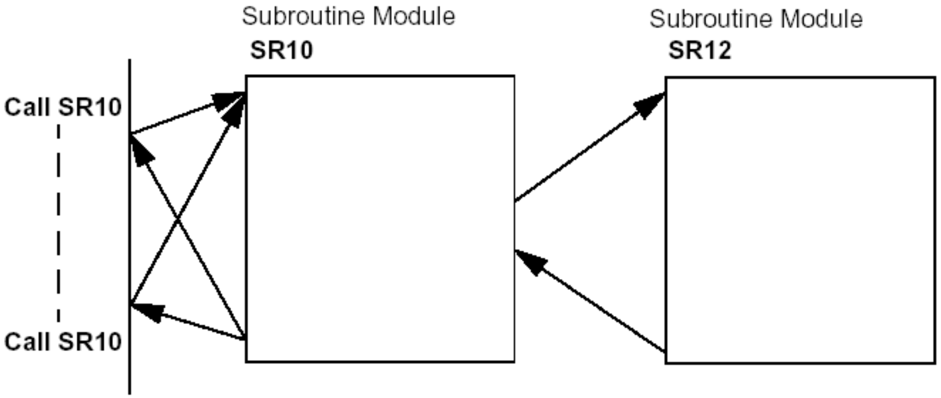
Rank	Instruction
1	Instruction to an operand
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

Instruction list

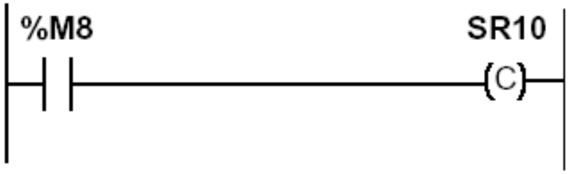
Structures for Control of Flux

Subroutines

Call and Return



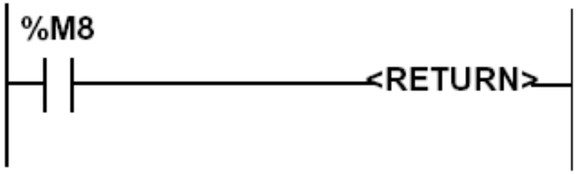
Ladder language:



Instruction list language:

```
LD   %M8
SR10
```

Ladder language



Instruction list language

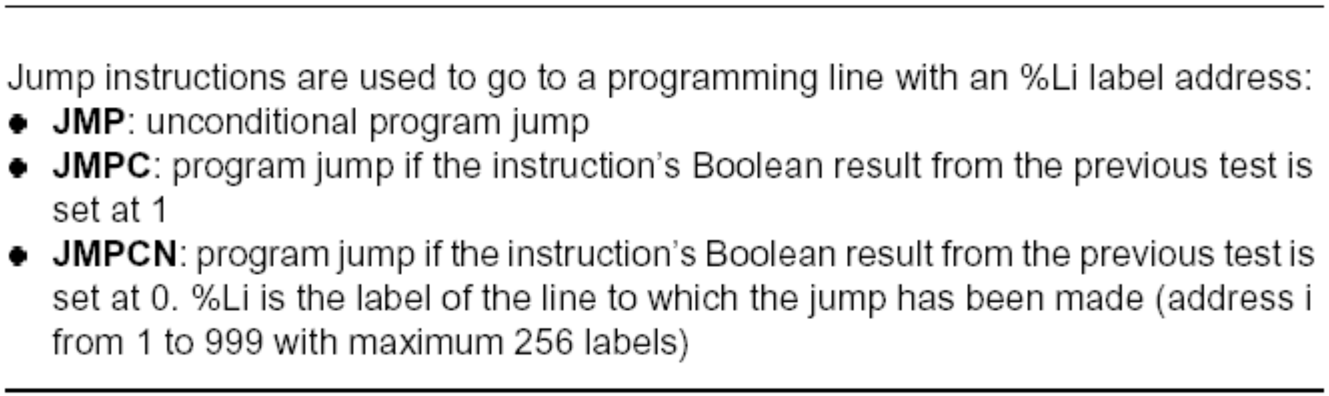
```
LD   %M8
RETC
```

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:

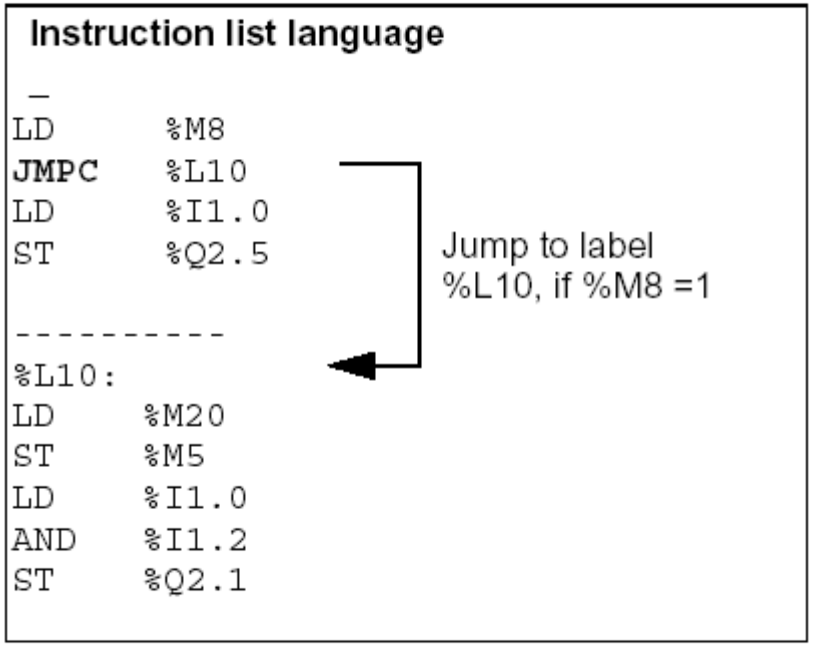
- **JMP**: unconditional program jump
- **JMPC**: program jump if the instruction's Boolean result from the previous test is set at 1
- **JMPCN**: program jump if the instruction's Boolean result from the previous test is set at 0. %Li is the label of the line to which the jump has been made (address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

Instruction list

Structures for Control of Flux

Example:

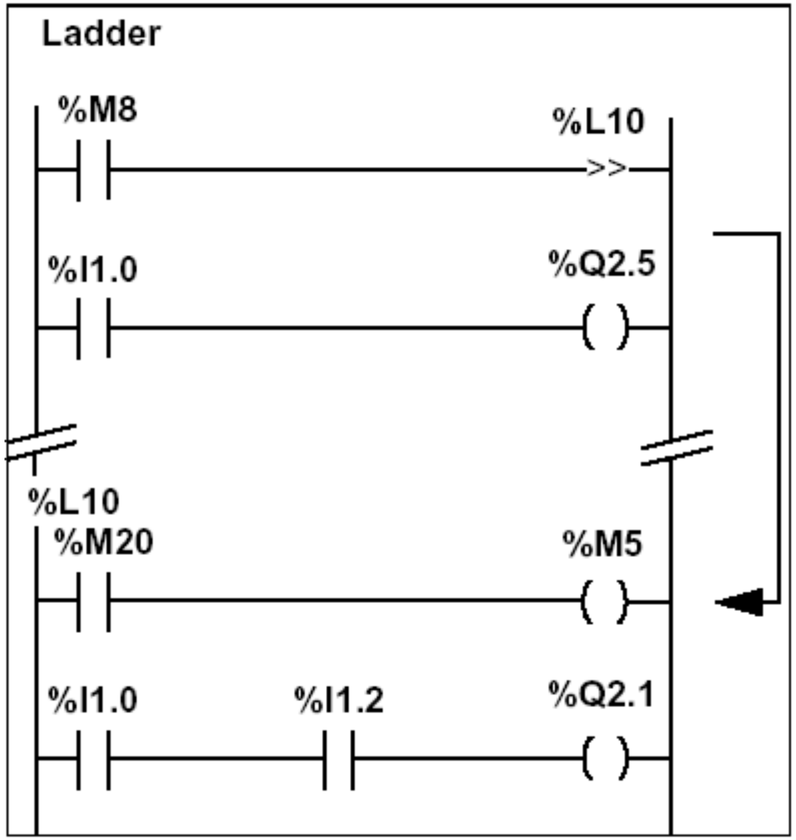
Use of jump instructions



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
```

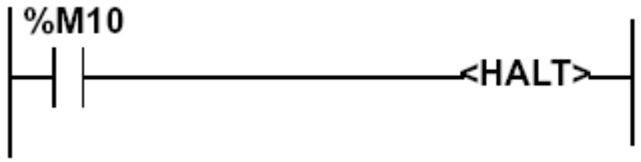
Jump to label
%L10, if %M8 =1



Instruction list

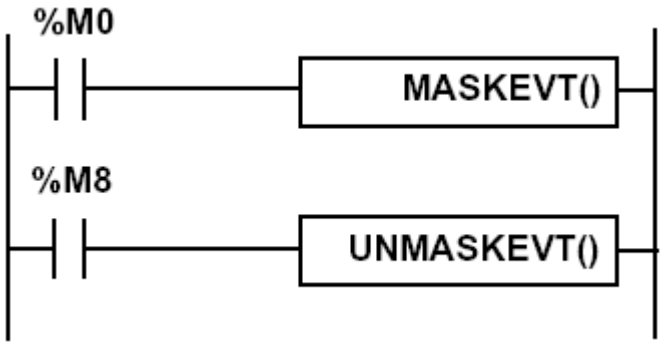
Structures for Control of Flux

Halt



Stops all processes!

Events masking



Instruction list

There are other advanced instrauctions (see manual)

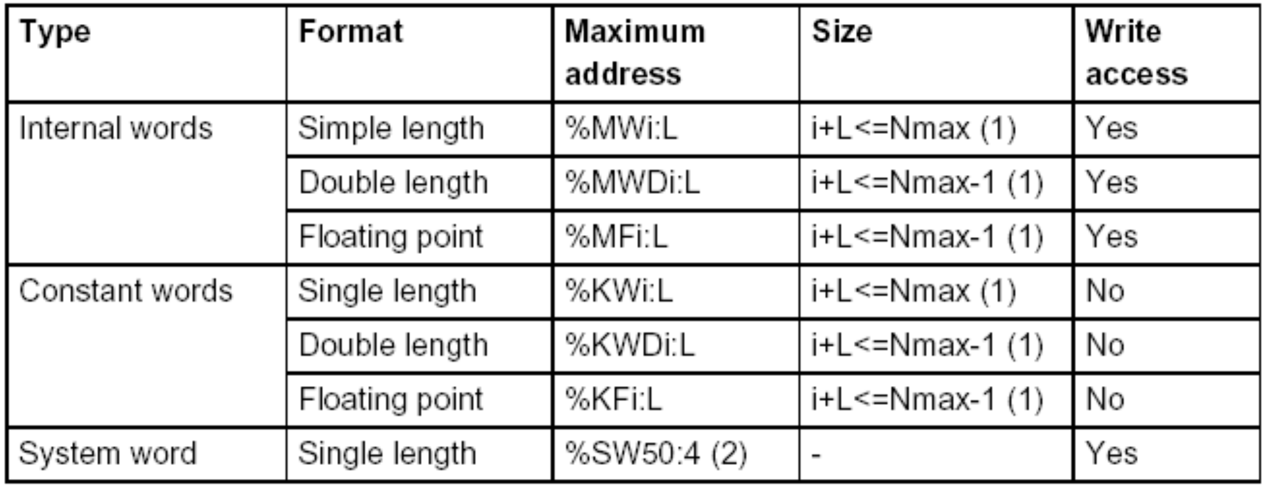
- **Monostable**
- **Registers of 256 words (LIFO ou FIFO)**
- ***DRUMs***
- **Comparators**
- ***Shift-registers***

...

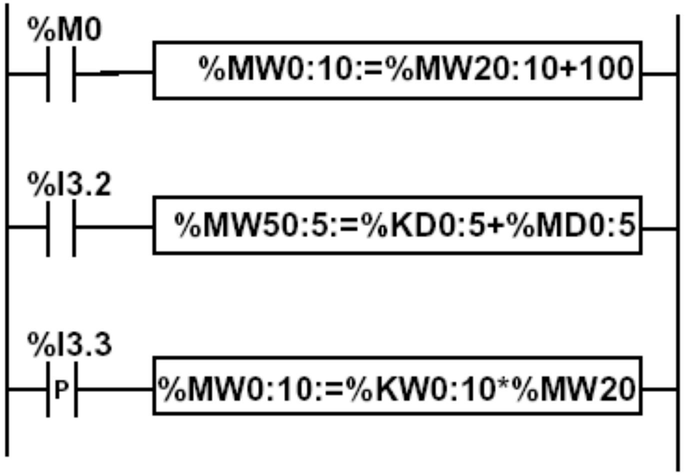
- **Functions to manipulate *floats***
- **Functions to convert bases and types**

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*

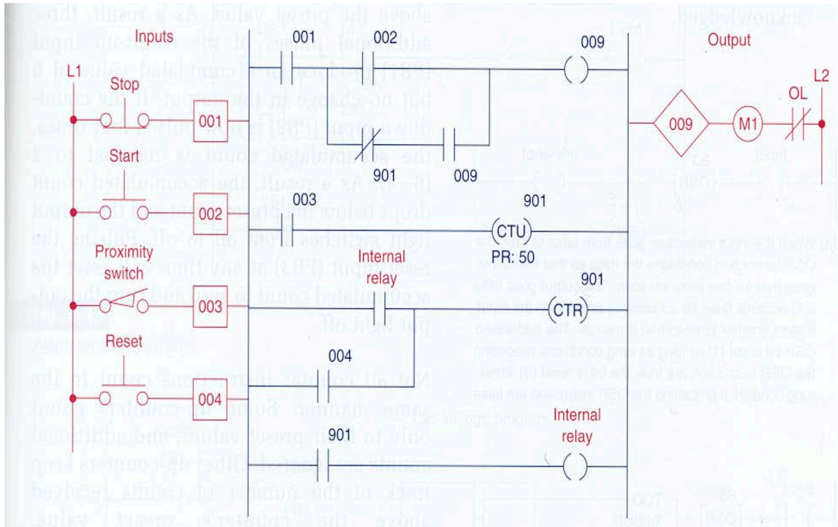
DOLOG80 (cont.)

Example:

```
AI1
A(
OI2
O(
ANC9
AQ9
)
)
=Q9
...
```

```
AI3
=P9
NO
OM1
OI4
=Z9
NO
AC9
=M1
...
```

```
LDV50
=CSW9
PE
```



(b) Program

Fig. 8-13 Conveyor motor program.

Legend:

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

Industrial Automation

(Automação de Processos Industriais)

PLCs Programming Languages
Ladder Diagram

http://www.isr.ist.utl.pt/~pjcro/courses/api0910/api0910.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.*

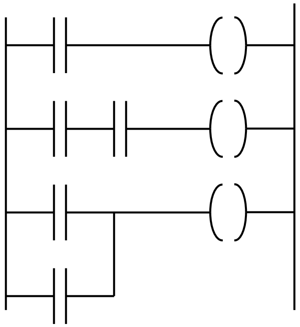
Software development resources.

...

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

PLCs Programming Languages (IEC 1131-3)

Ladder Diagram



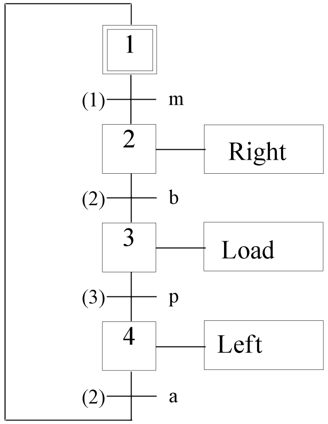
Structured Text

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

Instruction List

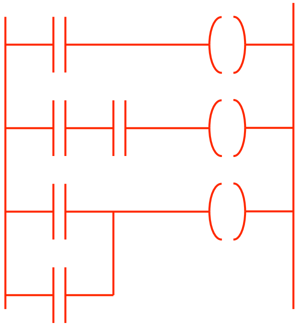
```
LD        %M12
AND       %I1.0
ANDN      %I1.1
OR        %M10
ST        %Q2.0
```

Sequential Function Chart (GRAFCET)



Linguagens de programação de PLCs (IEC 1131-3)

Ladder Diagram



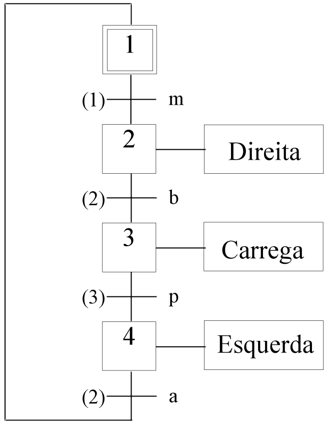
Structured Text

```
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
```

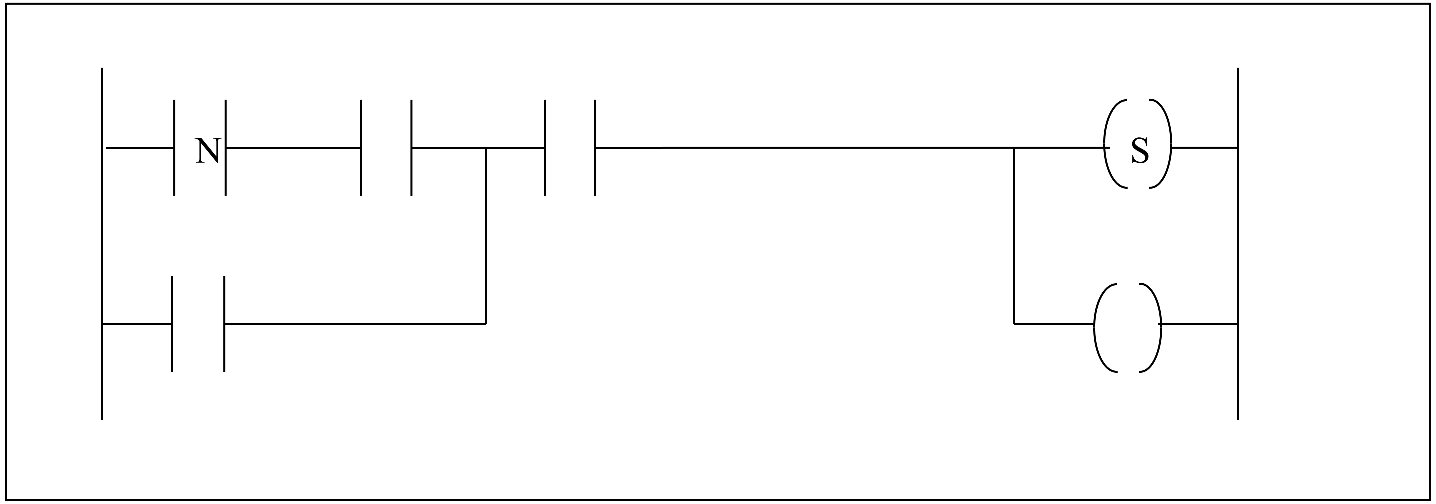
Sequential Function Chart (GRAFCET)



Ladder diagram

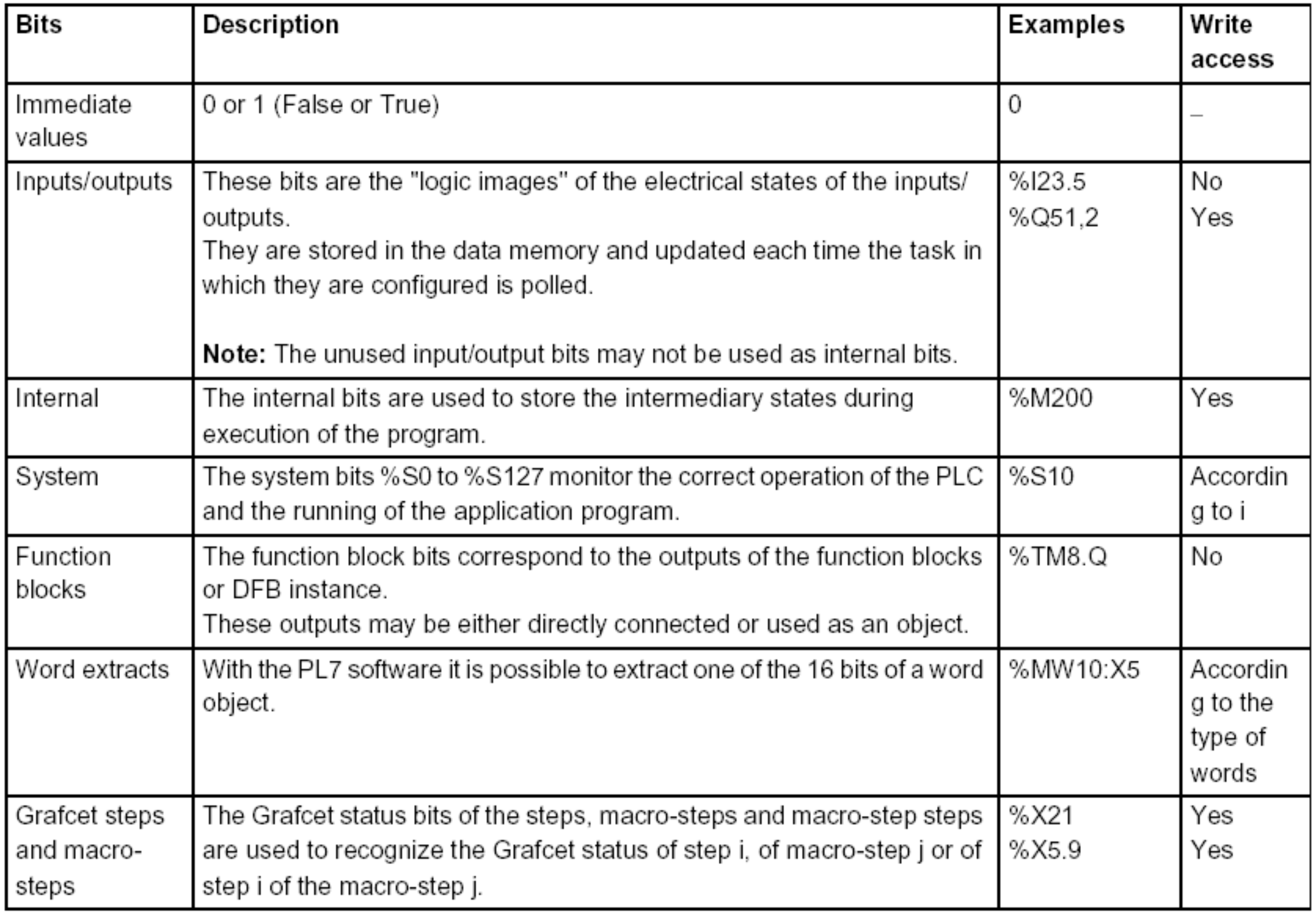
Input Instructions

Output Instructions



Ladder diagram

Types of operands:

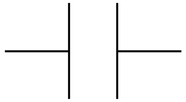


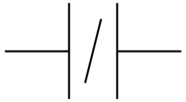
Bits	**Description**	**Examples**	**Write access**
Immediate values	0 or 1 (False or True)	0	_
Inputs/outputs	These bits are the "logic images" of the electrical states of the inputs/ outputs. They are stored in the data memory and updated each time the task in which they are configured is polled. **Note:** 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

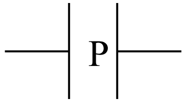
Ladder diagram

Basic Instructions

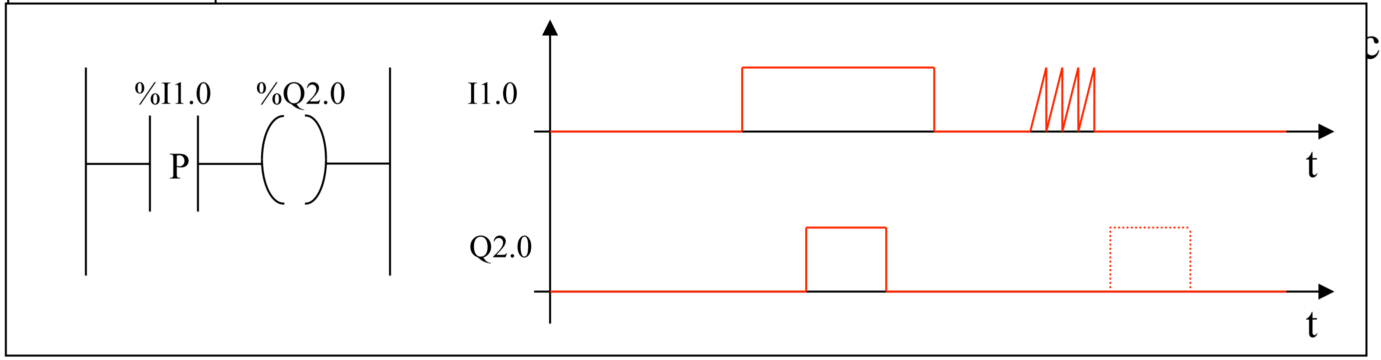
Load


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.

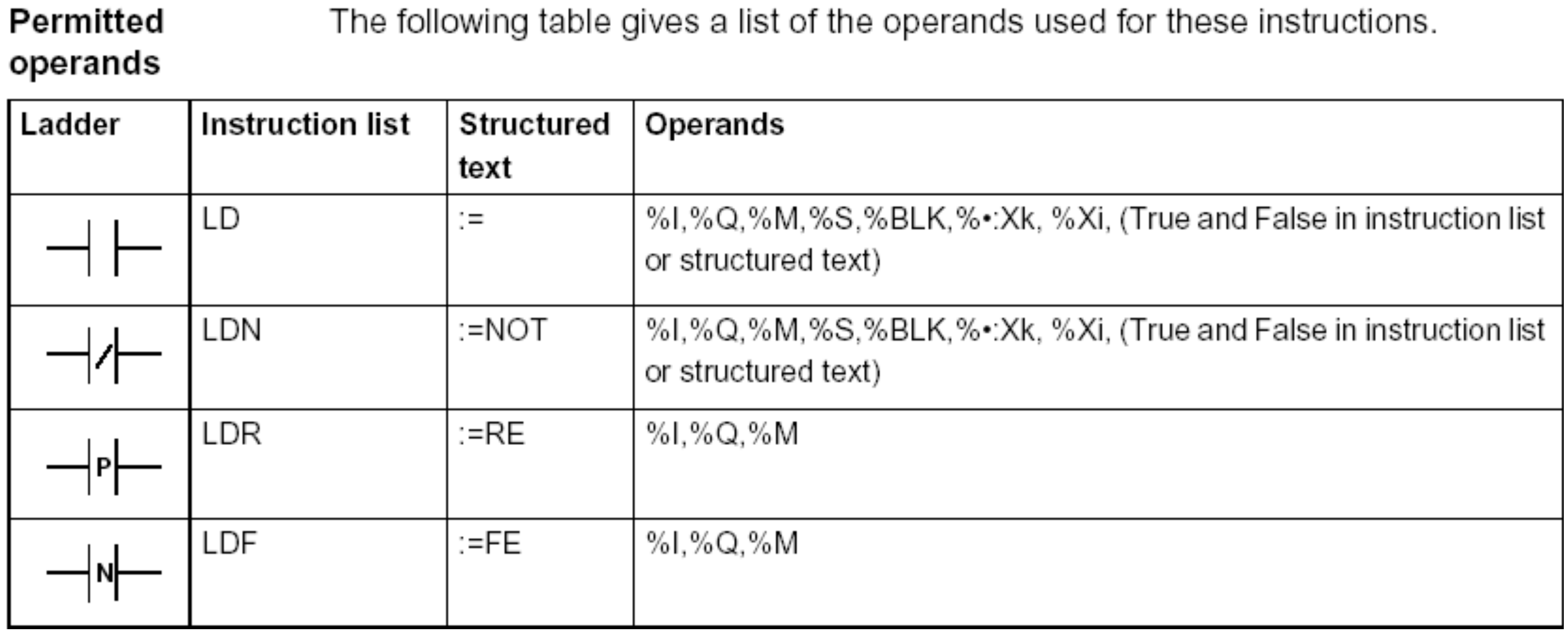


cycle

Ladder diagram

Basic Instructions

Load operands



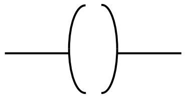
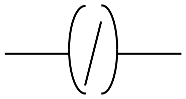
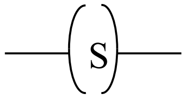
Permitted operands The following table gives a list of the operands used for these instructions.

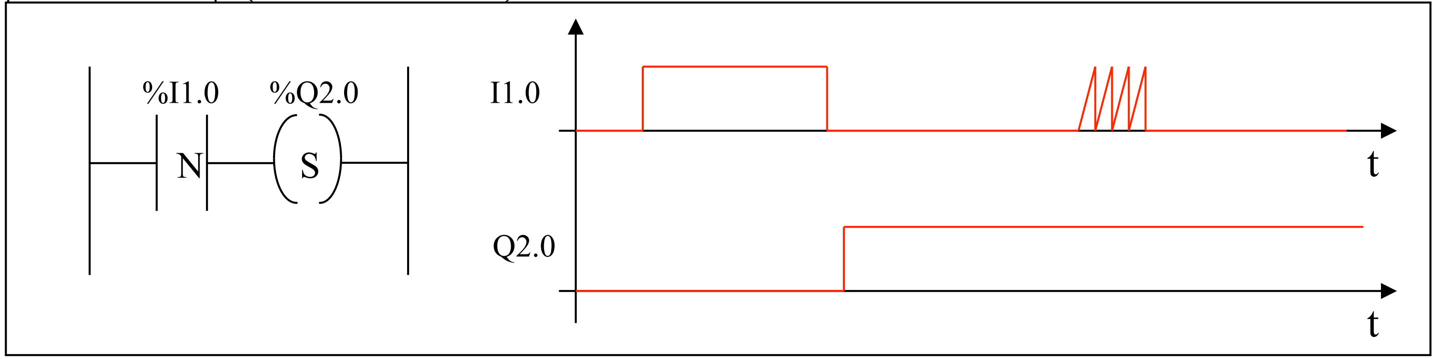
Ladder	Instruction list	Structured text	Operands
—| |—	LD	:=	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi, (True and False in instruction list or structured text)
—|/|—	LDN	:=NOT	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi, (True and False in instruction list or structured text)
—|P|—	LDR	:=RE	%I,%Q,%M
—|N|—	LDF	:=FE	%I,%Q,%M

Ladder diagram

Basic Instructions

Store

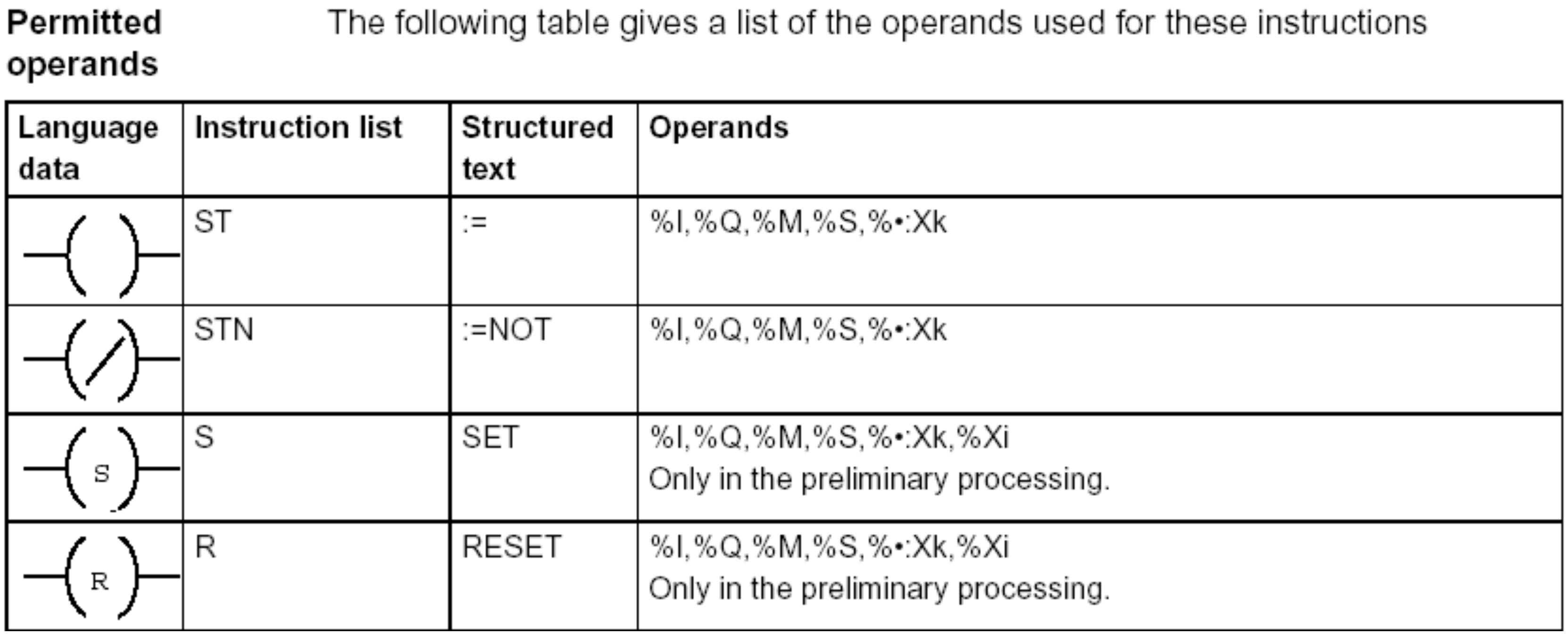
	The result of the logic function activates the coil.
	The inverse result of the logic function activates the coil.
	The result of the logic function energizes the relay (sets the latch).



Ladder diagram

Basic Instructions

Store operands



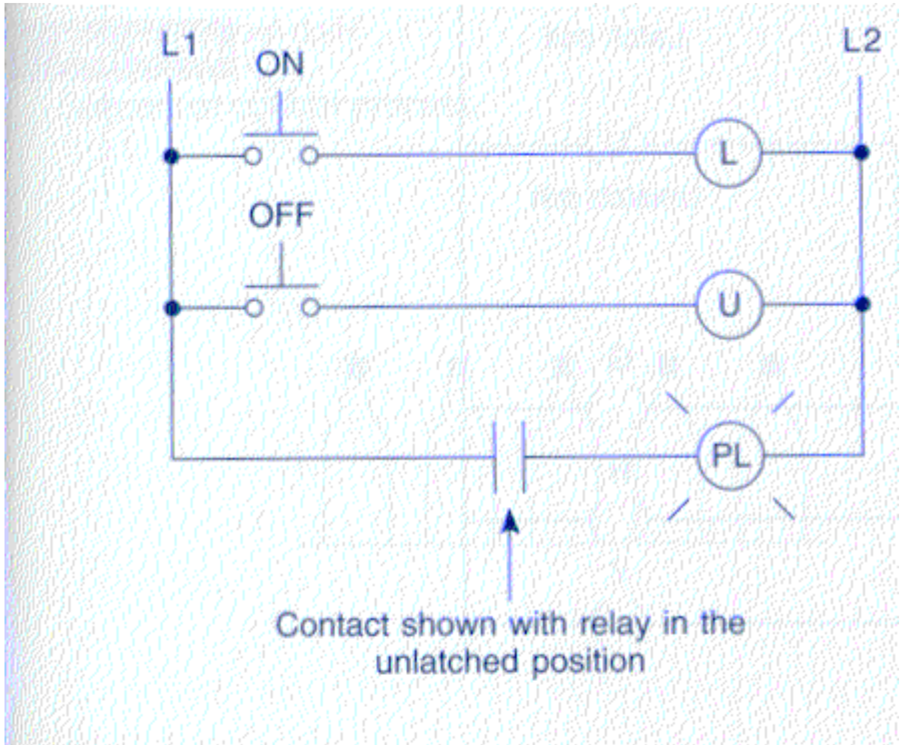
Permitted operands The following table gives a list of the operands used for these instructions

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*

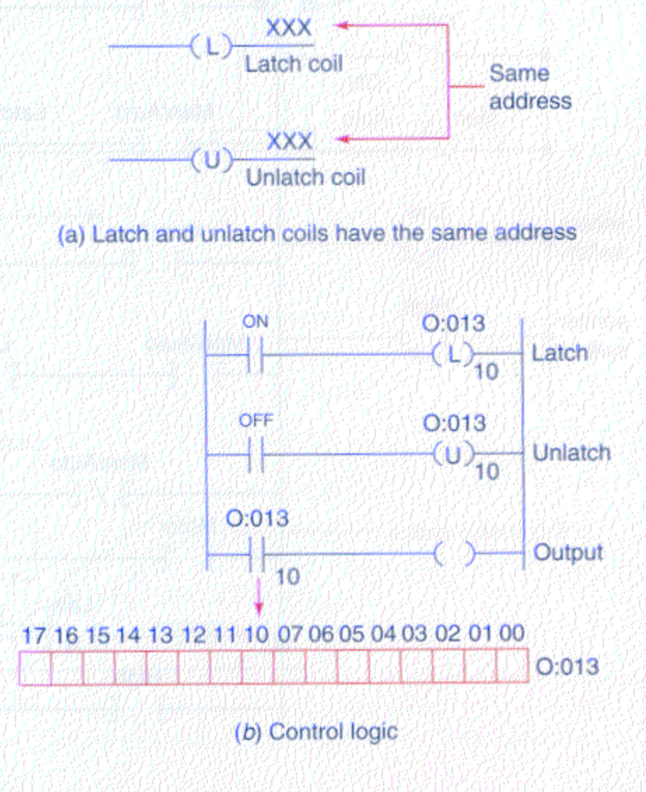


Contact shown with relay in the unlatched position

Fig. 6-50

Schematic of electromagnetic latching relay.

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



(a) Latch and unlatch coils have the same address

(b) Control logic

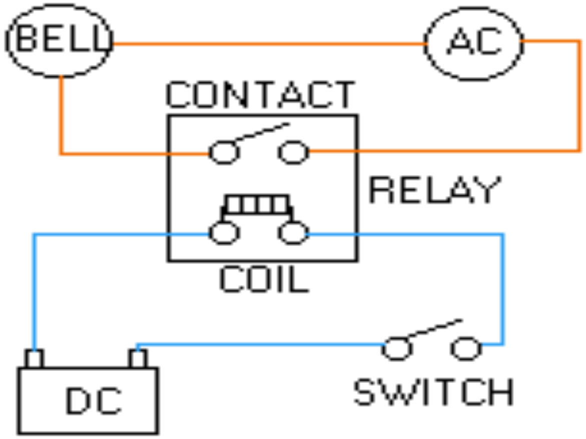
Fig. 6-51

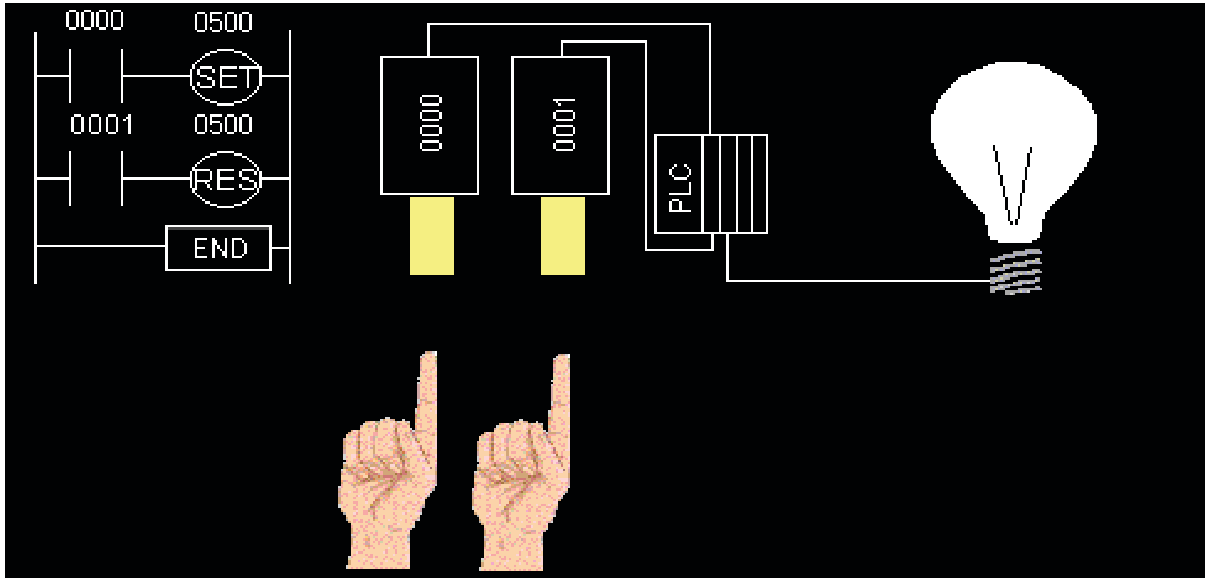
OUTPUT LATCH and OUTPUT UNLATCH instructions.

Ladder diagram

Relay-type instructions

Example:

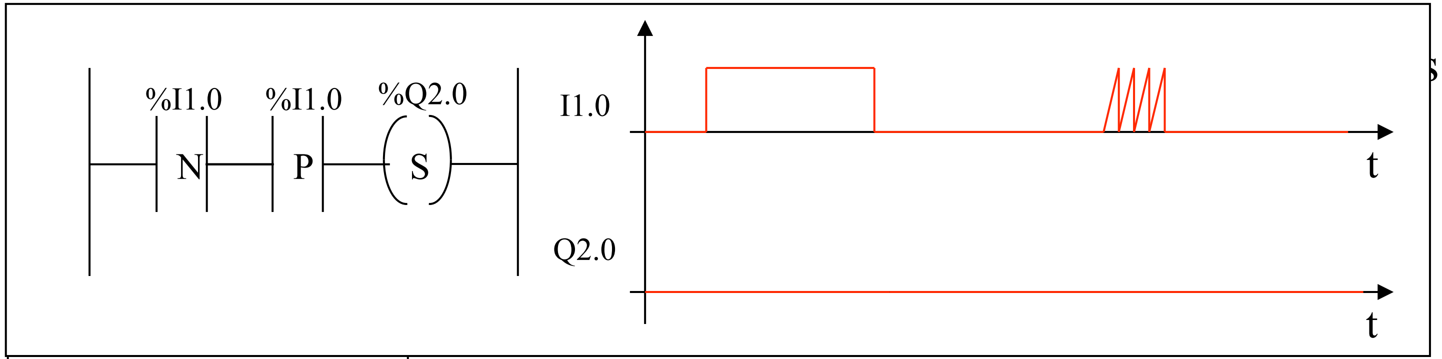




Ladder diagram

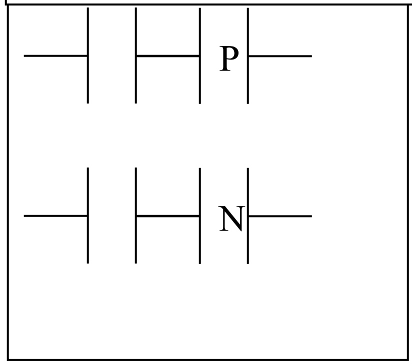
Basic Instructions

AND



s

previous



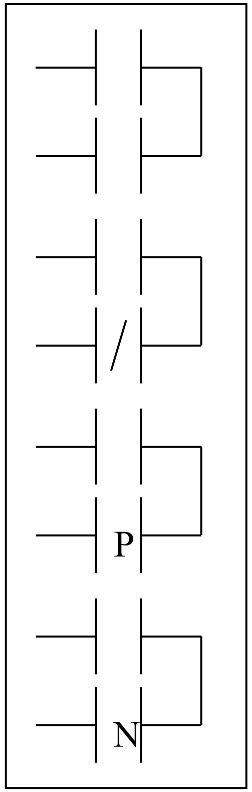
AND of the rising edge with the result of the previous logical operation.

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

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.

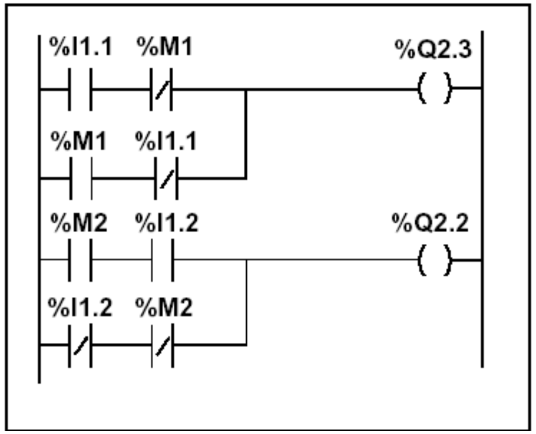
OR of the rising edge with the result of the previous logical operation.

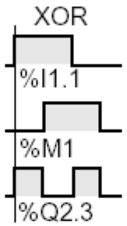
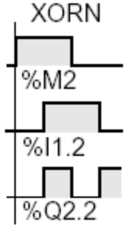
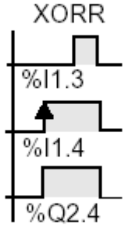
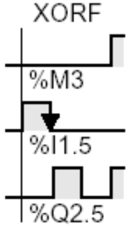
OR of the falling edge with the result of the previous logical operation.

Ladder diagram

Basic Instructions

XOR



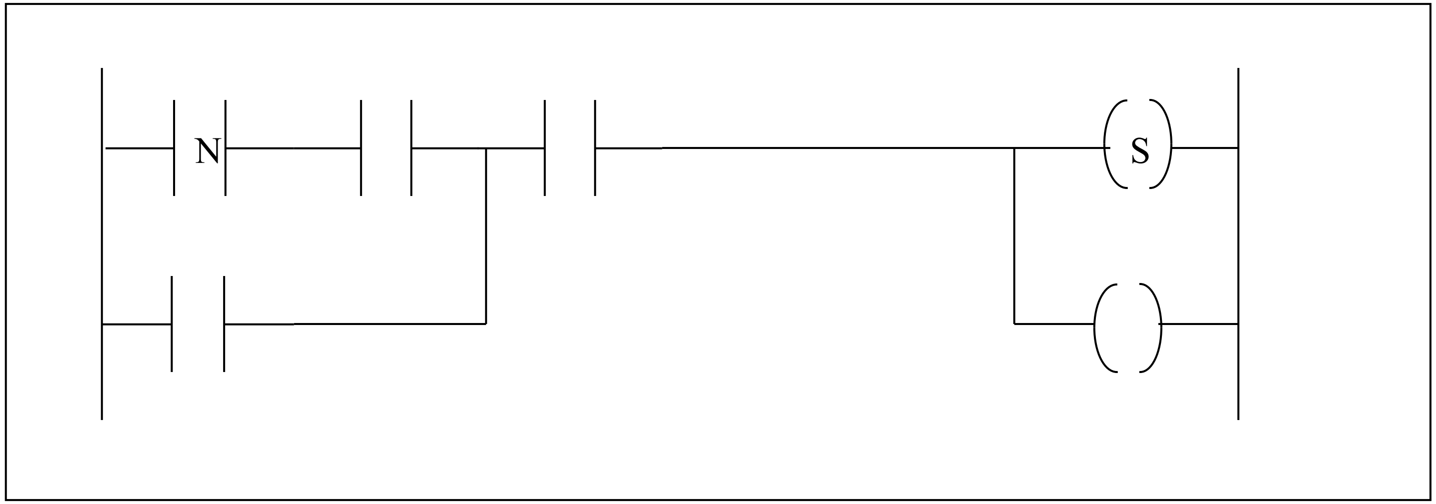
Instruction list	Structured text	Description	Timing diagram
XOR	XOR	OR Exclusive between the operand and the previous instruction's Boolean result	
XORN	XOR (NOT...)	OR Exclusive between the operand inverse and the previous instruction's Boolean result	
XORR	XOR (RE...)	OR Exclusive between the operand's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

Ladder diagram

Ladder assembling

Input Instructions

Output Instructions



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:

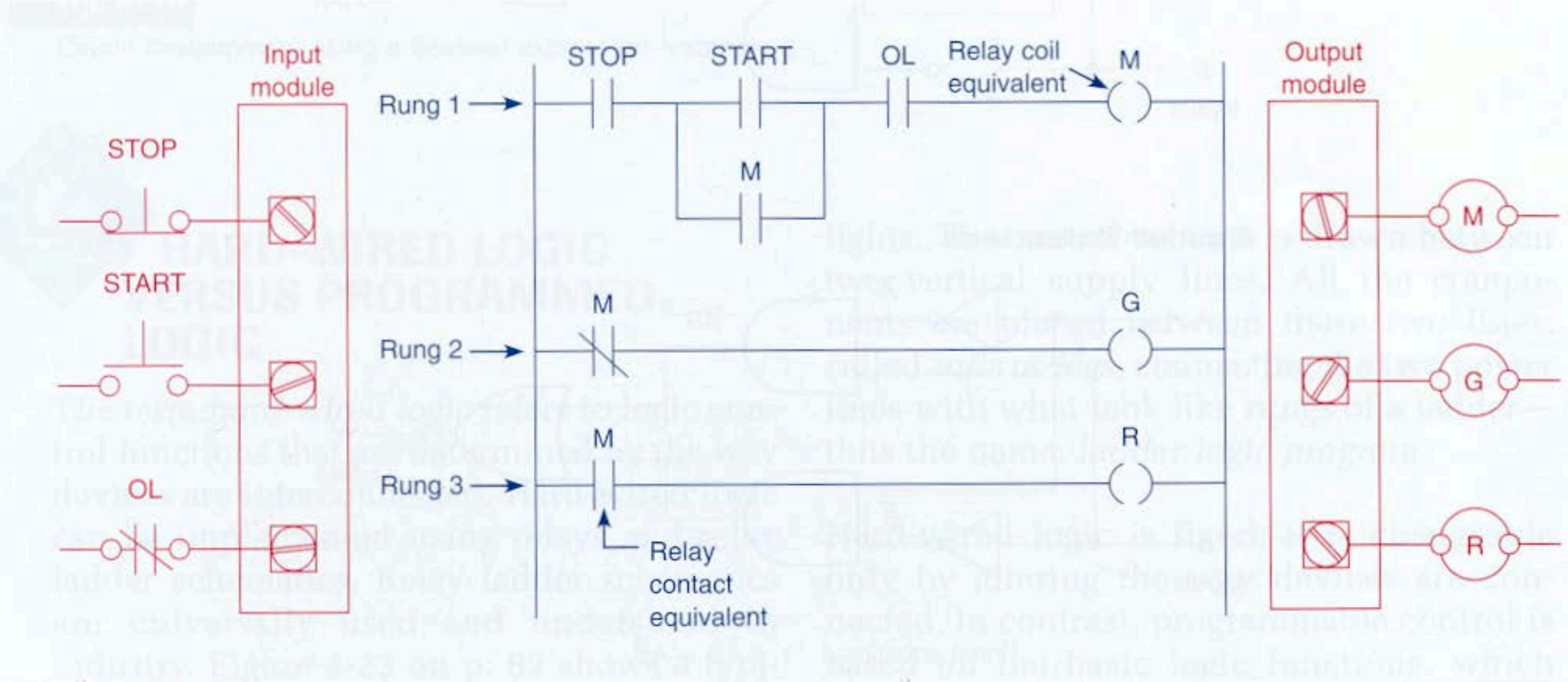


Fig. 4-24

Ladder logic program.

Ladder diagram

Example:

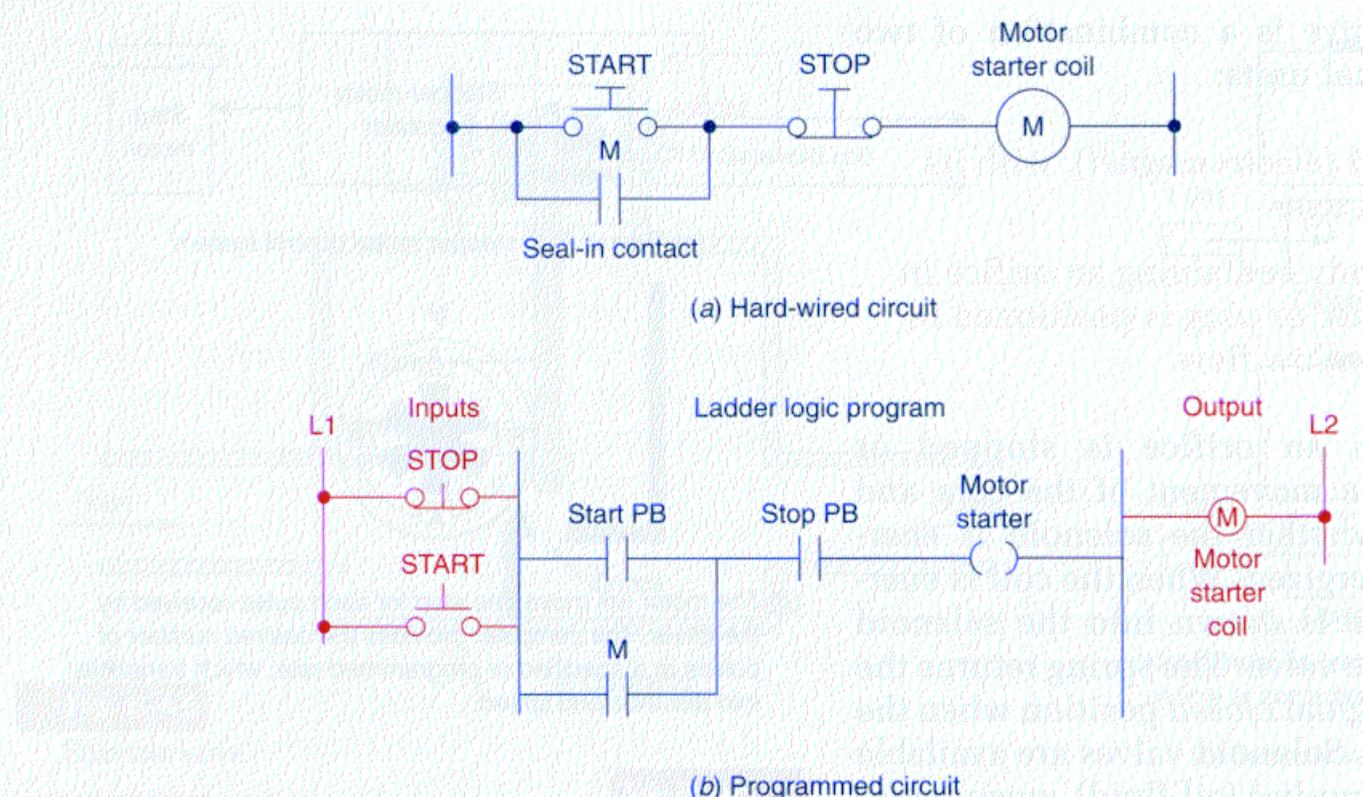


Fig. 6-48
Seal-in circuit.

Ladder diagram

Example:

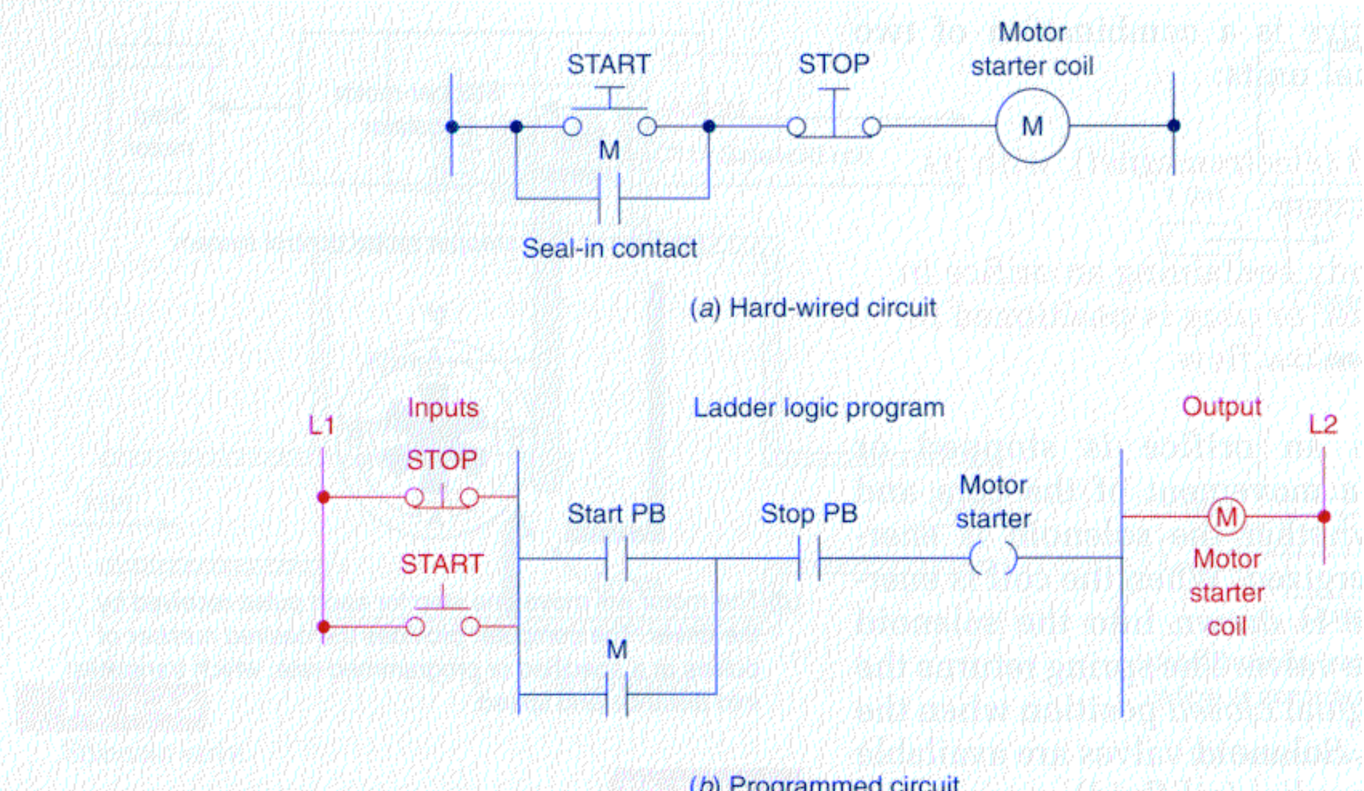
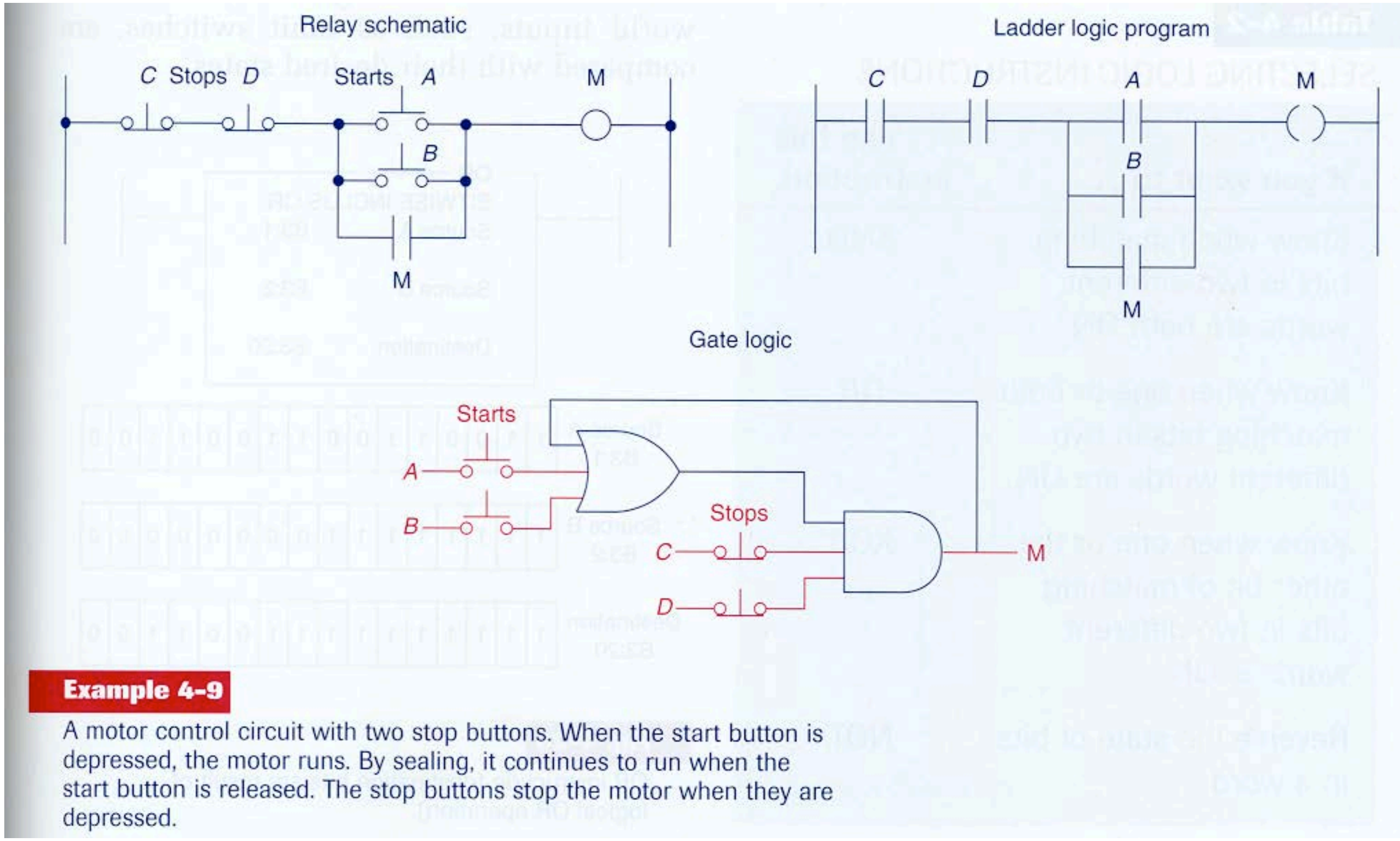


Fig. 6-48
Seal-in circuit.

Ladder diagram

Example:



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

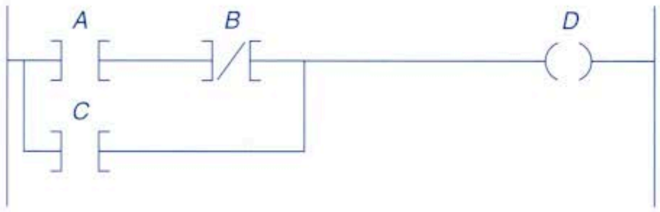


Fig. 5-21
Parallel input branching.

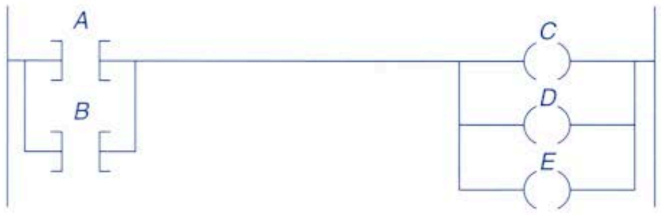


Fig. 5-22
Parallel output branching.

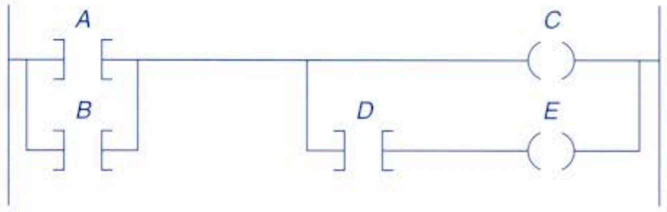


Fig. 5-23
Parallel output branching with conditions.

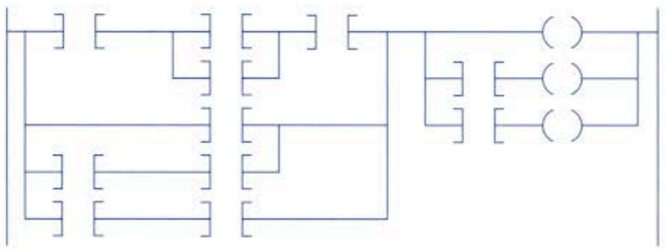


Fig. 5-24
Nested input and output branches.

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

Ladder diagram

Imbricated contacts and alternative solution

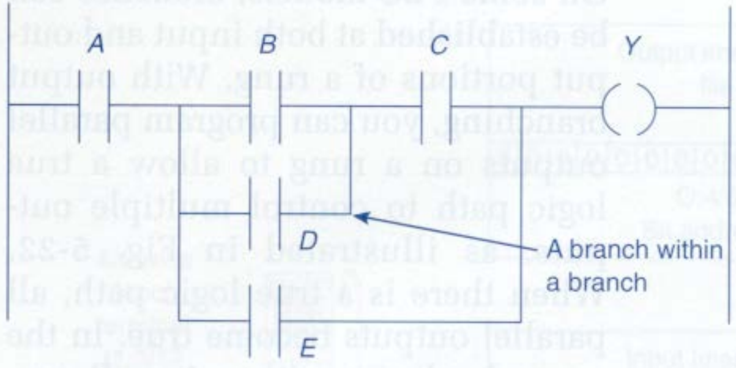


Fig. 5-25
Nested contact program.

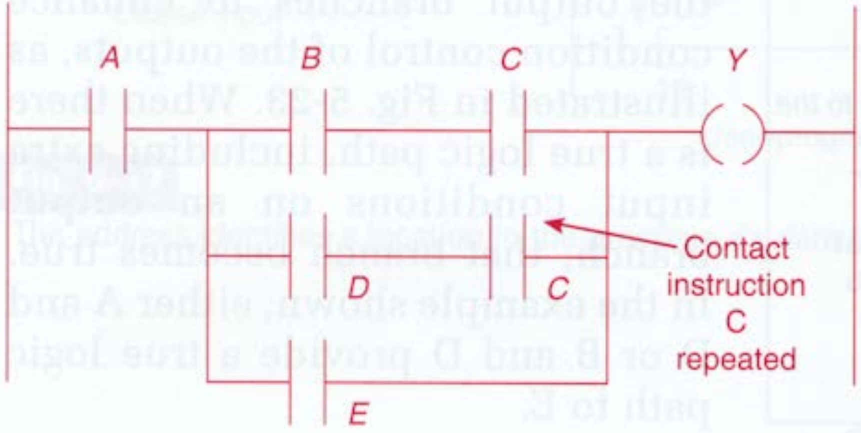
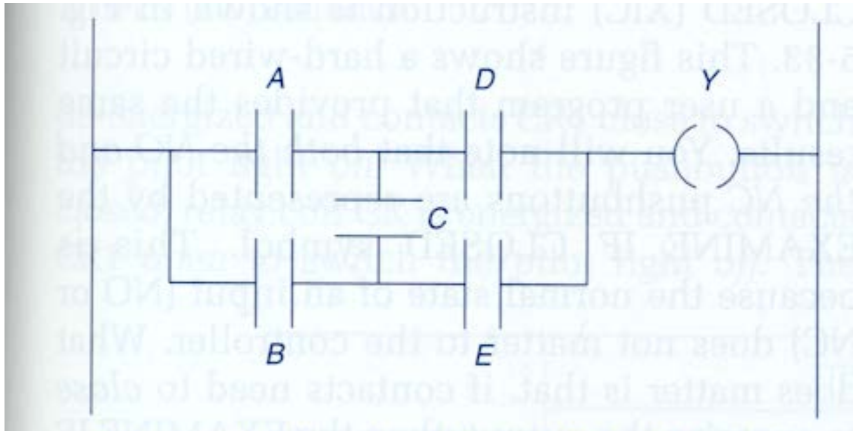


Fig. 5-26
Program required to eliminate nested contact.

Ladder diagram

Contacts in the vertical and alternative solution



Boolean equation: $Y = (AD) + (BCD) + (BE) + (ACE)$

Fig. 5-28

Program with vertical contact

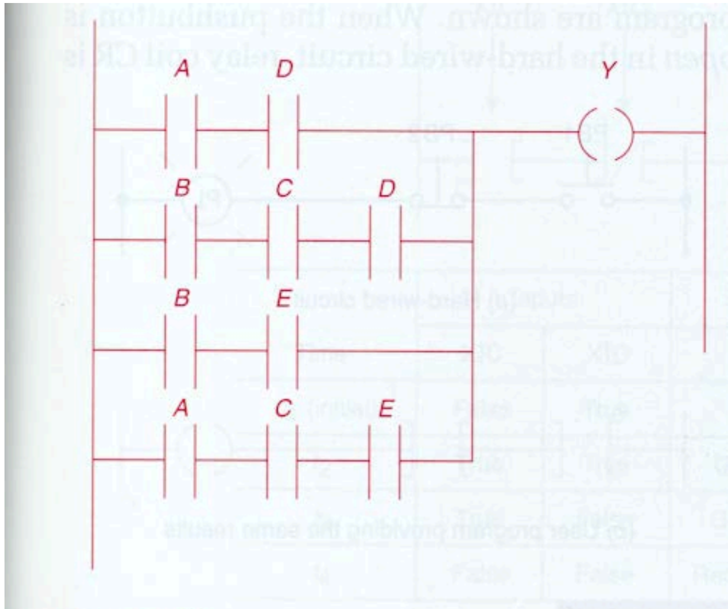


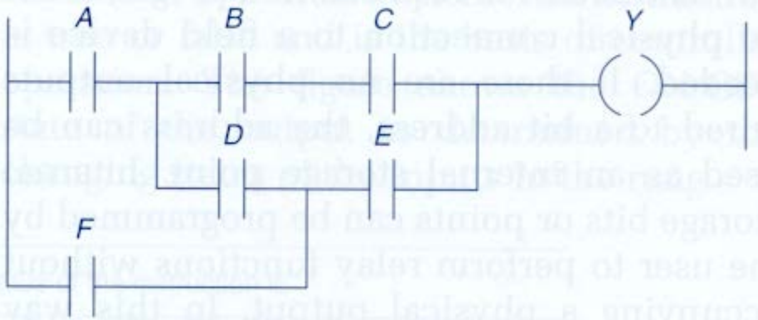
Fig. 5-29

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.

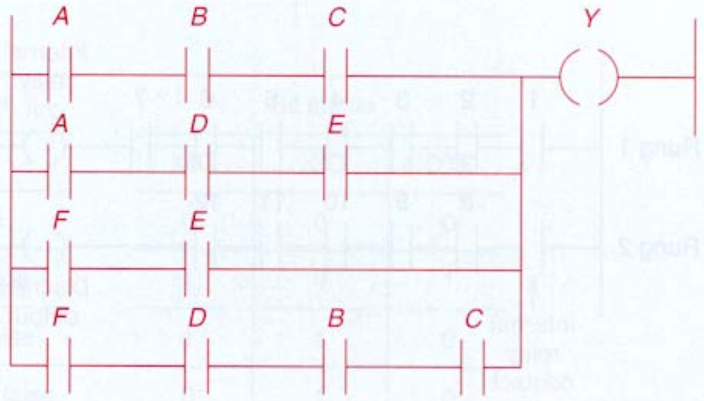


Fig. 5-31

Reprogrammed circuit.

Ladder diagram

Temporized Relays

or

Timers

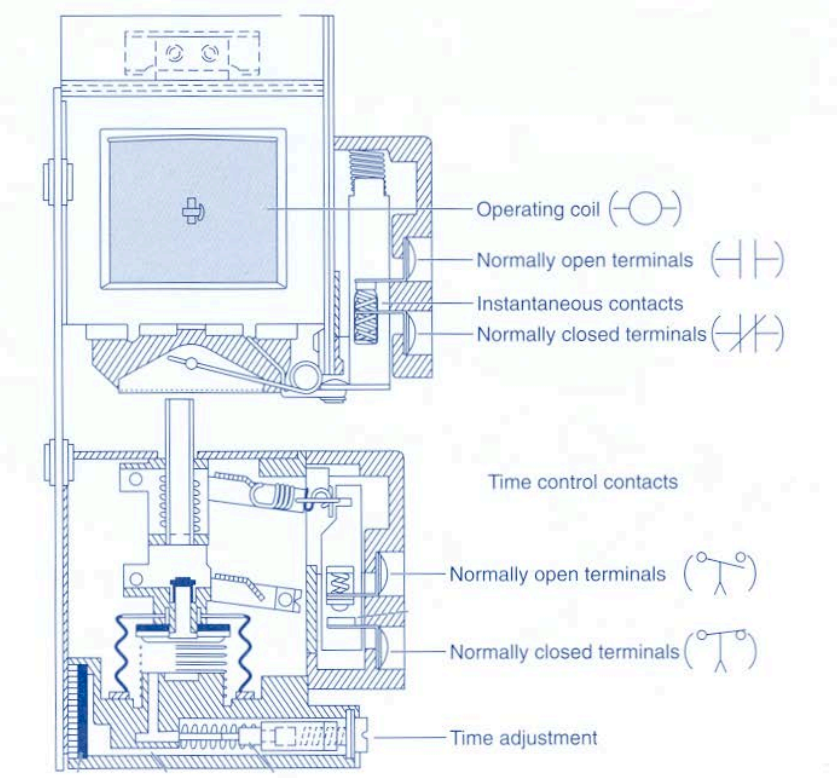
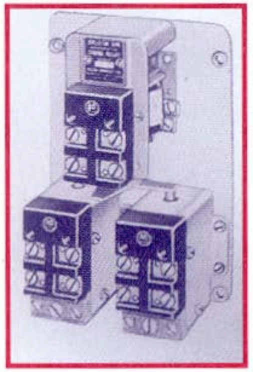


Fig. 7-1

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

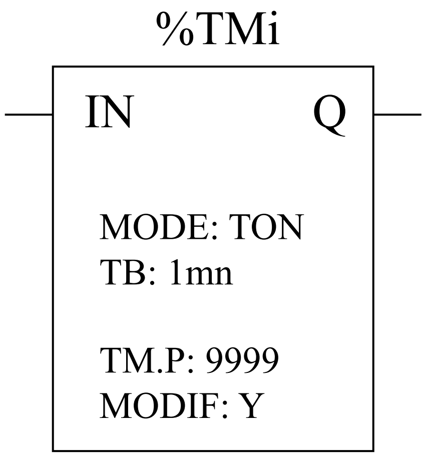


Ladder diagram

Temporized Relays

or

Timers



Characteristics:

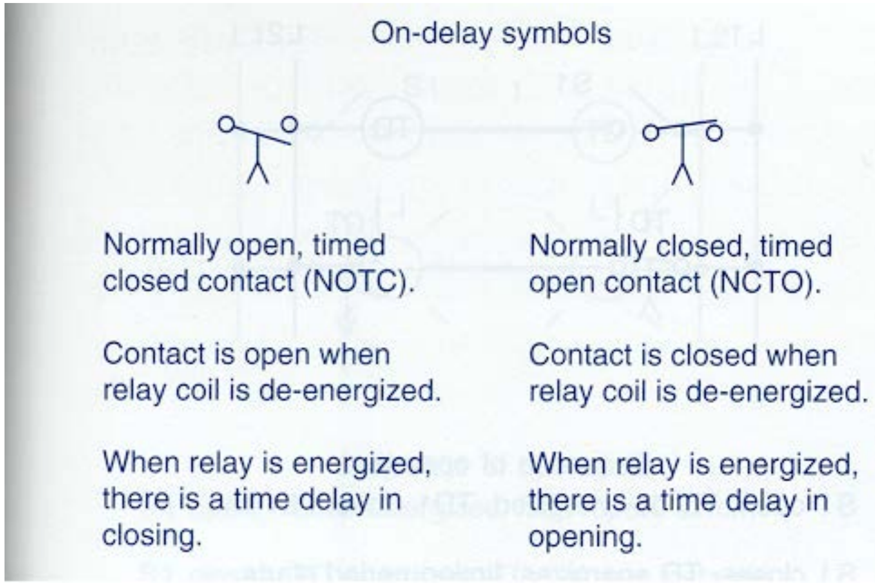
Identifier:%TMi	0..63 in the TSX37	
Input:	IN	to activate
Mode:	TON TOFF TP	On delay Off delay Monostable
Time basis:	TB	1mn (def.), 1s, 100ms, 10ms
Programmed value:	%TMi.P	0...9999 (def.) period=TB*TMi.P
Actual value:	%TMi.V	0...TMi.P (can be real or tested)
Modifiable:	Y/N	can be modified from the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

Ladder diagram

Temporized Relays

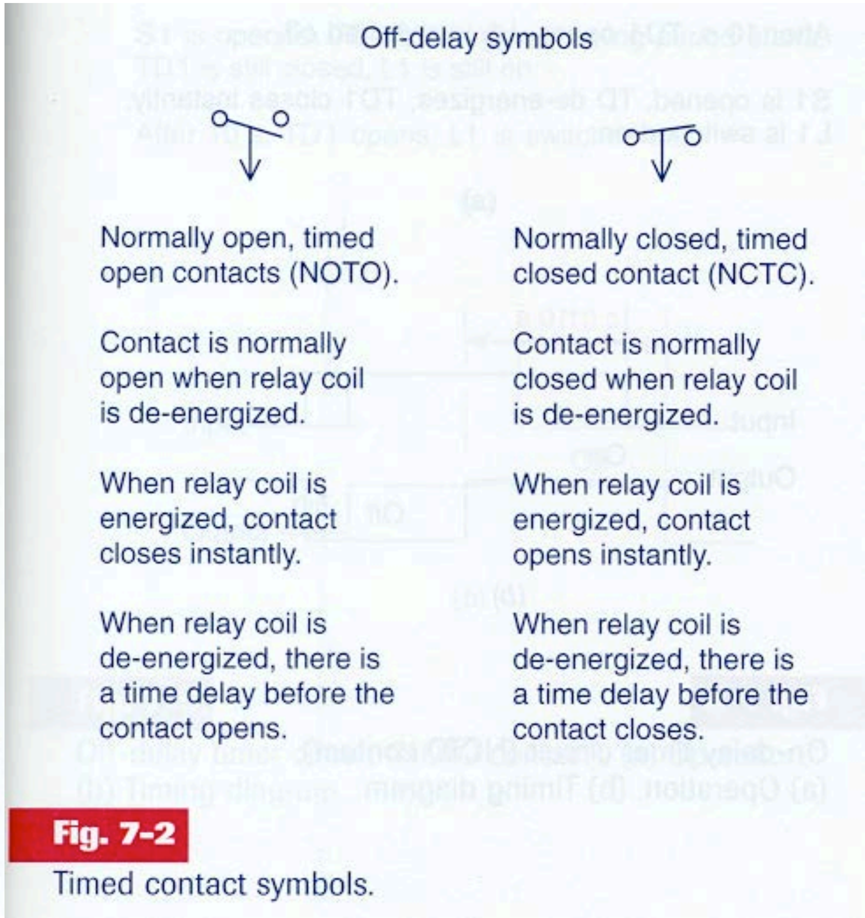
or

Timers



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

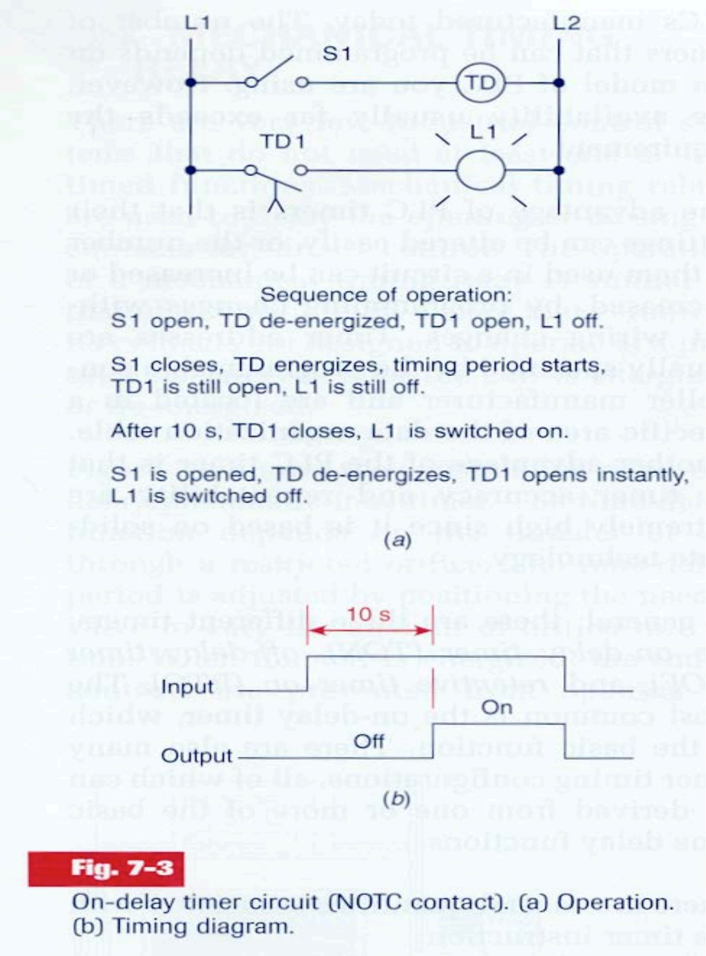
Normally open, timed open contacts (NOTO).	Normally closed, timed closed contact (NCTC).
Contact is normally open when relay coil is de-energized.	Contact is normally closed when relay coil is de-energized.
When relay coil is energized, contact closes instantly.	When relay coil is energized, contact 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.

Fig. 7-2

Timed contact symbols.

Ladder diagram

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.

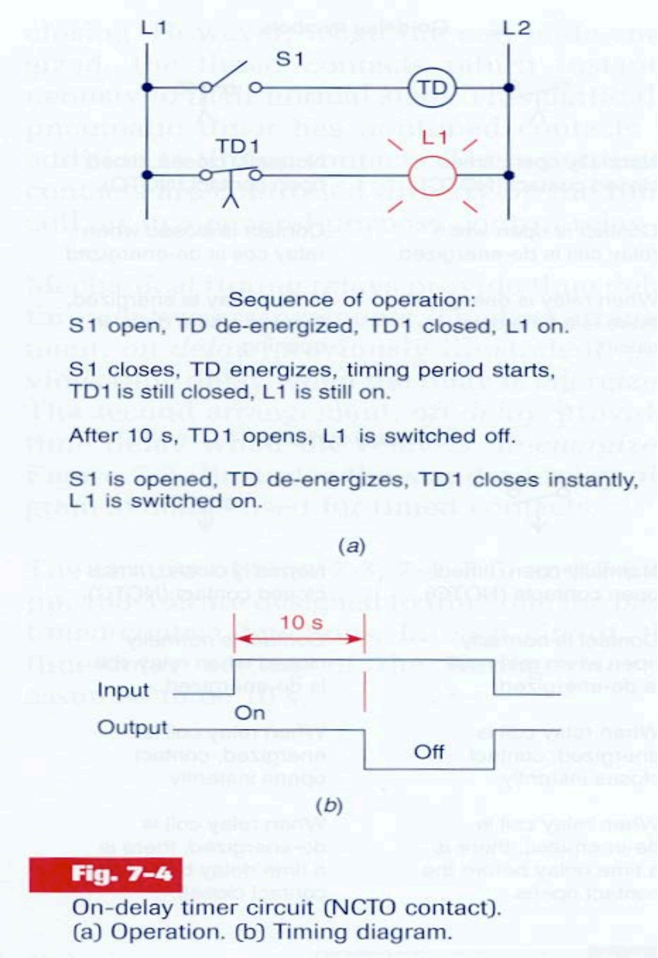
S1 is opened, TD de-energizes, TD1 opens instantly, L1 is switched off.

(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)

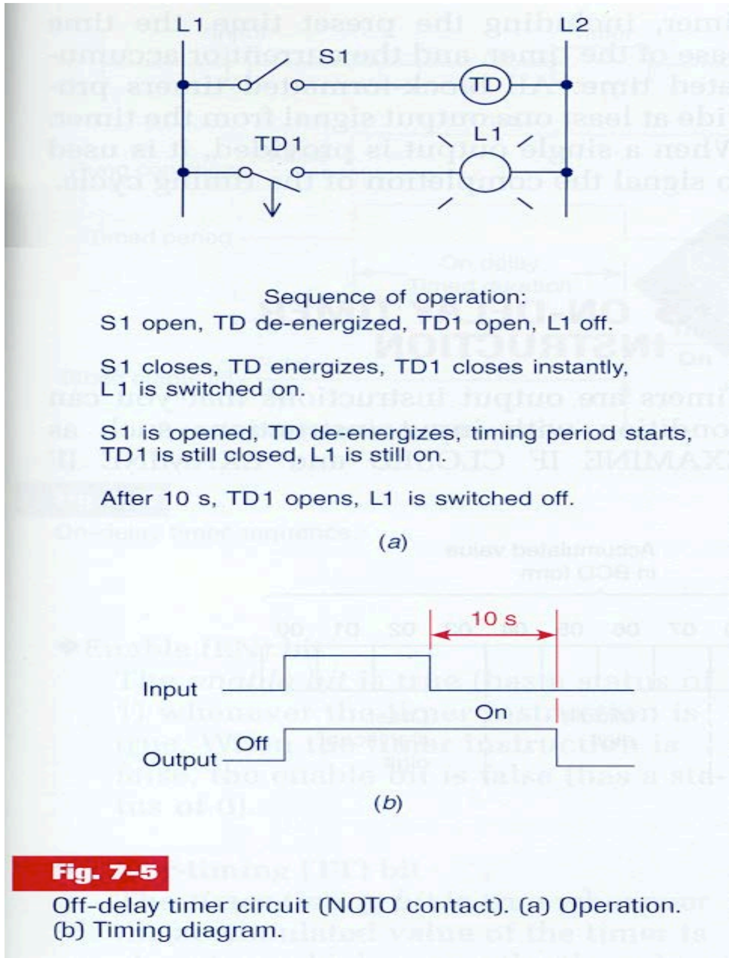
(b)

Fig. 7-4

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

Ladder diagram

Example:



Sequence of operation:

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

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

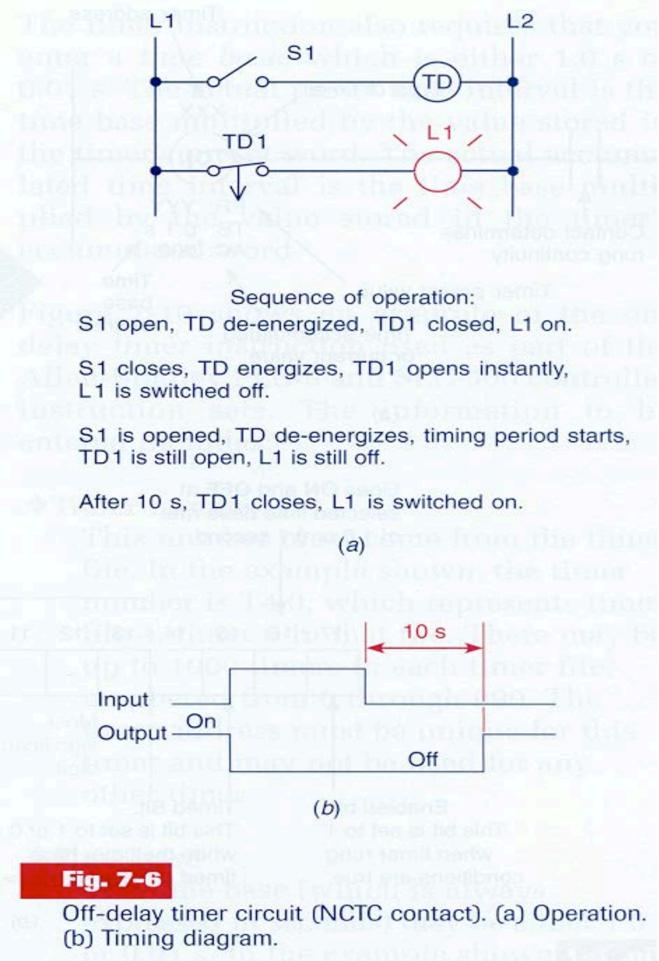
S1 is opened, TD de-energizes, timing period starts, TD1 is still closed, L1 is still on.

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*)

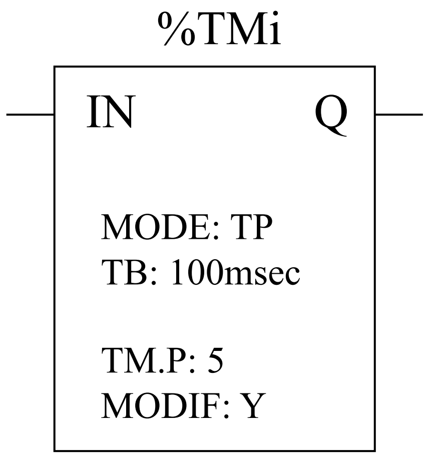
Fig. 7-6 Off-delay timer circuit (NCTC contact). (a) Operation. (b) Timing diagram.

Ladder diagram

Temporized Relays

or

Timers

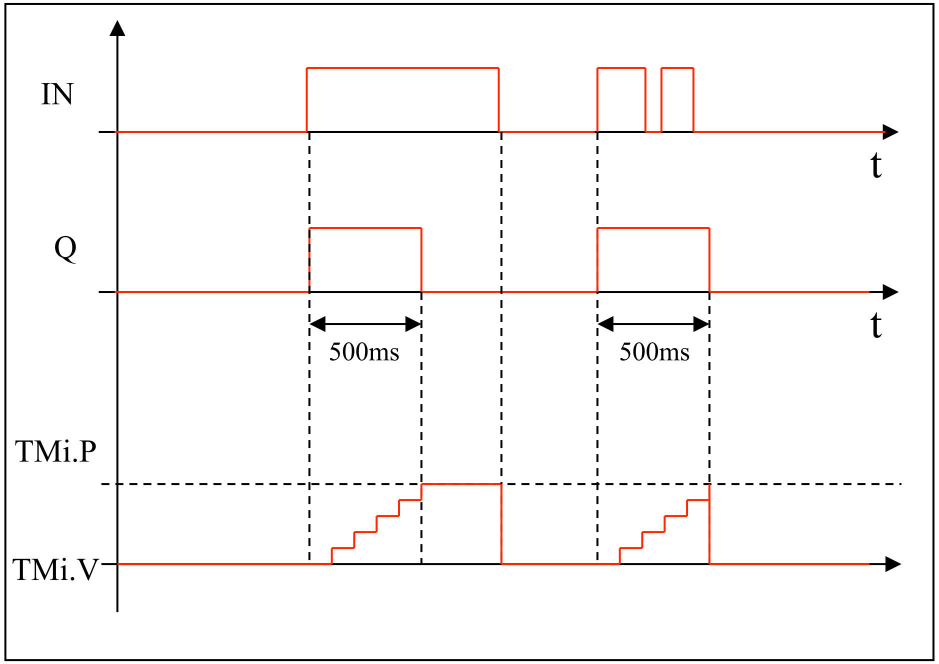


MODE: TP
TB: 100msec

TM.P: 5
MODIF: Y

Mode: TP

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



Ladder diagram

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

Two alternative representations....

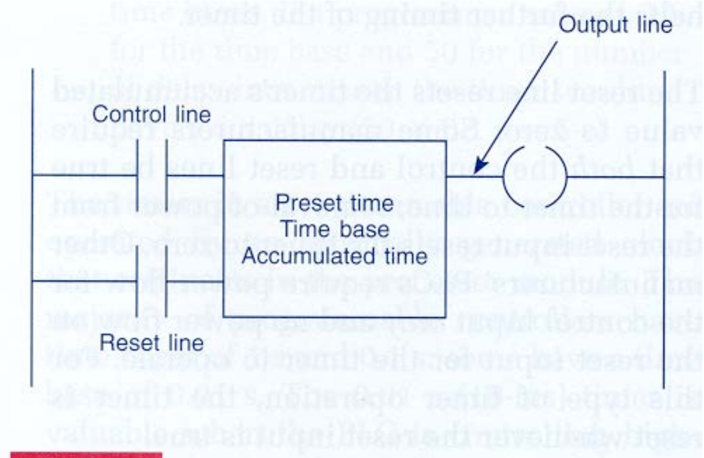


Fig. 7-8

Block-formatted timer instruction.

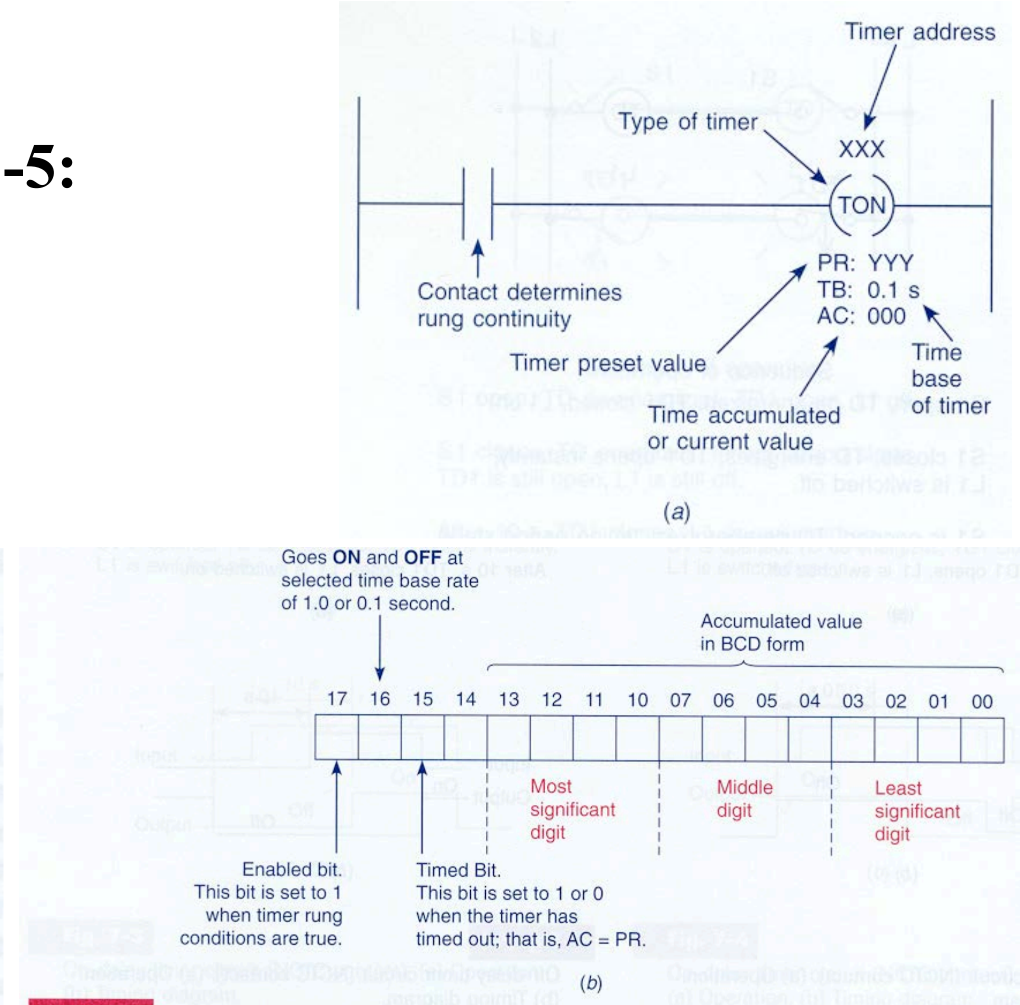

Timer address
Type of timer
XXX
TON
PR: YYY
TB: 0.1 s
AC: 000
Contact determines rung continuity
Timer preset value
Time base of timer
Time accumulated or current value
(a)
Goes ON and OFF at selected time base rate of 1.0 or 0.1 second.
Accumulated value in BCD form
17 16 15 14 13 12 11 10 07 06 05 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.
(b)

Fig. 7-7

Coil-formatted timer instruction. (a) Generic instruction. (b) Allen-Bradley PLC-2 timer accumulated value word.

Ladder diagram

Timers operation in the *Allen-Bradley* PLC-5

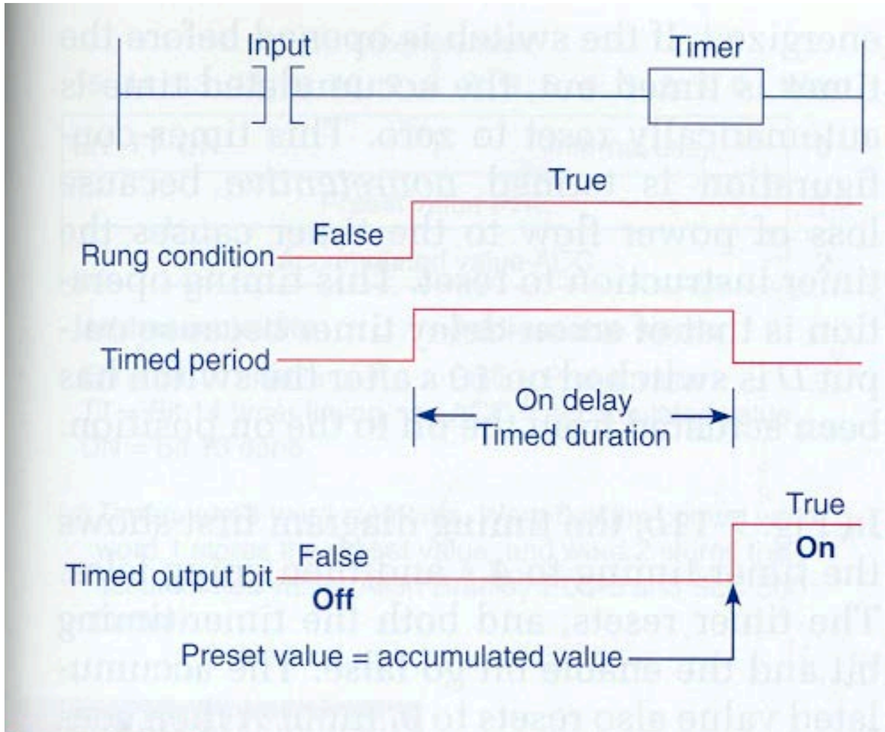


Fig. 7-9

On-delay timer sequence.

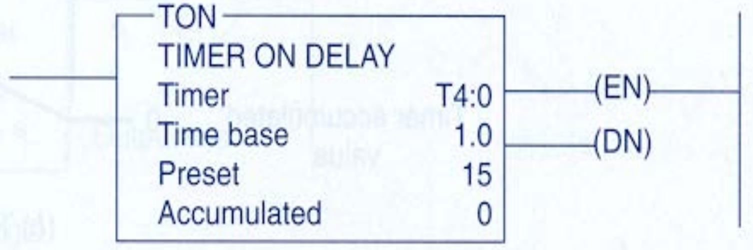


Fig. 7-10

On-delay timer instruction.

Ladder diagram

Example of *timer on-delay*

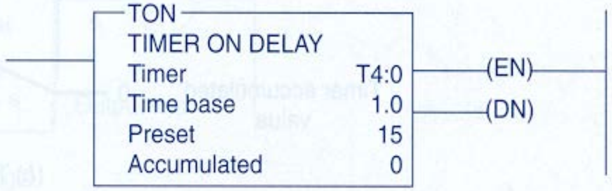


Fig. 7-10

On-delay timer instruction.

Timer element

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
EN	TT	DN		Internal use												0
Preset value PRE																1
Accumulated value ACC																2

Addressable bits	Addressable words
EN = Bit 15 enable	PRE = Preset value
TT = Bit 14 timer timing	ACC = Accumulated value
DN = Bit 13 done	

(*c*) Timers are 3-word elements. Word 0 is the control word, word 1 stores the preset value, and word 2 stores the accumulated value (Allen-Bradley PLC-5 and SLC-500 format).

Fig. 7-11 (continued)

On-delay timer.

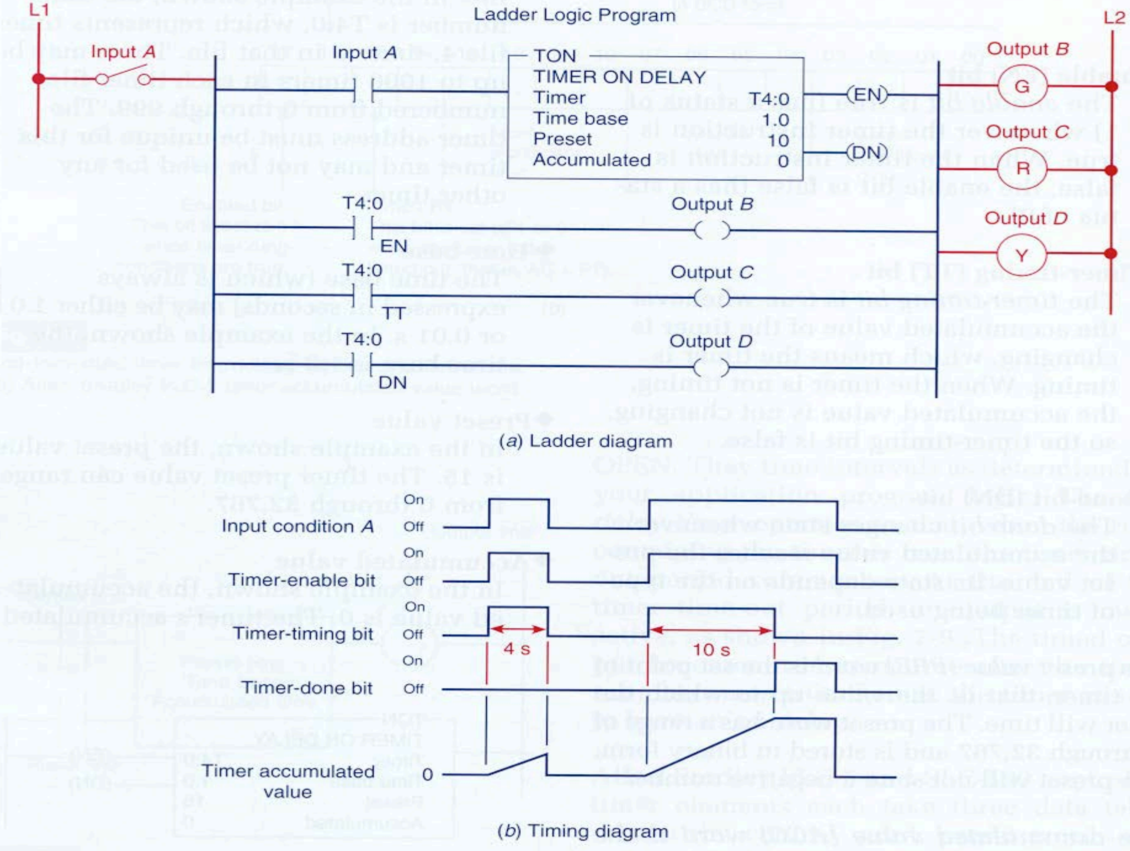
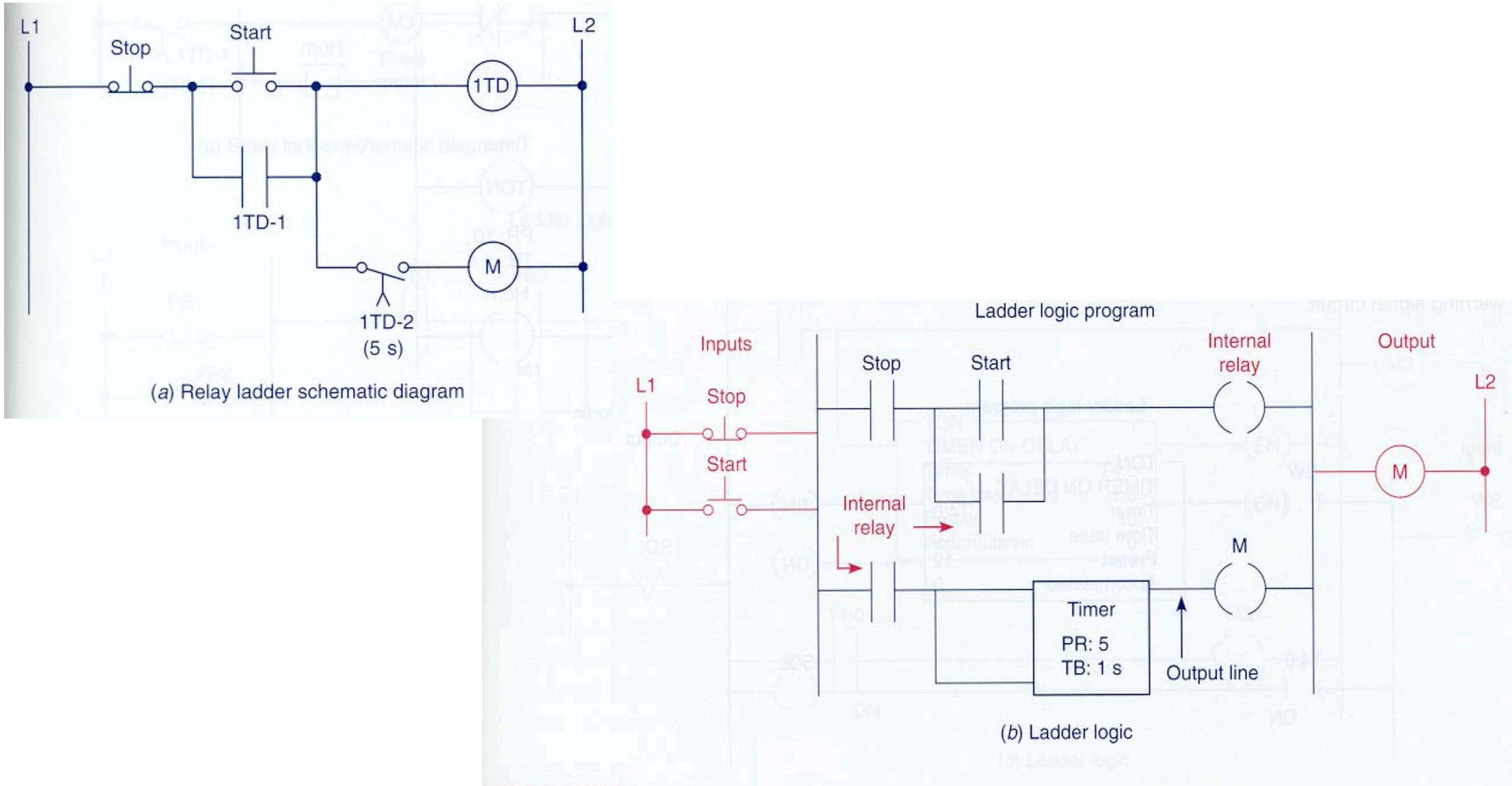


Fig. 7-11

On-delay timer.

Ladder diagram

Example of a *timer on-delay that sets an output*



(*a*) Relay ladder schematic diagram

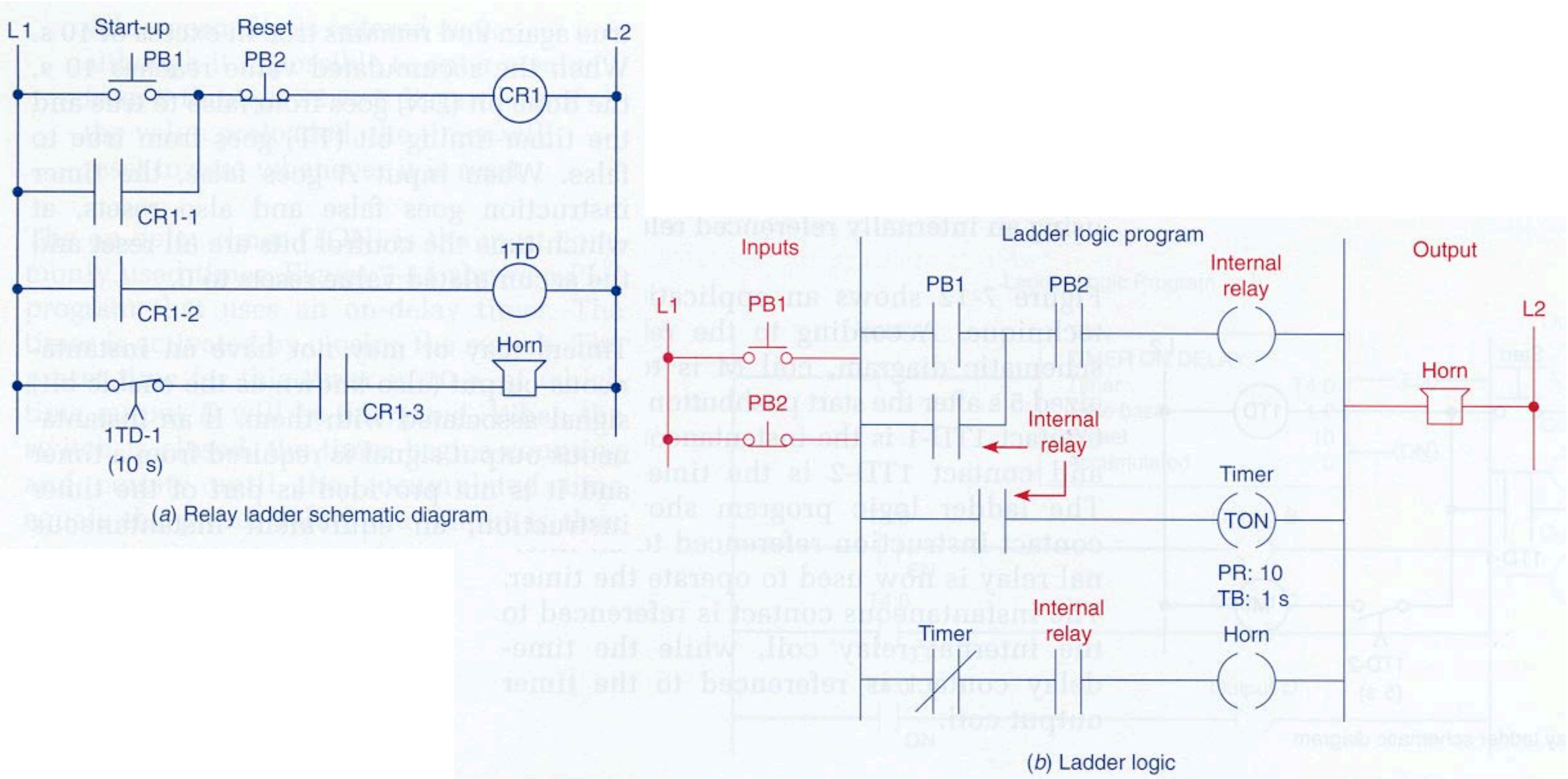
(*b*) Ladder logic

Fig. 7-12

On-delay timer with instantaneous output programming.

Ladder diagram

Example of *timer on-delay*



(*a*) Relay ladder schematic diagram

(*b*) Ladder logic

Fig. 7-13

Starting-up warning signal circuit.

Ladder diagram

Example of *timer on-delay*

Coil is energized if the switch remains closed for 12 seconds

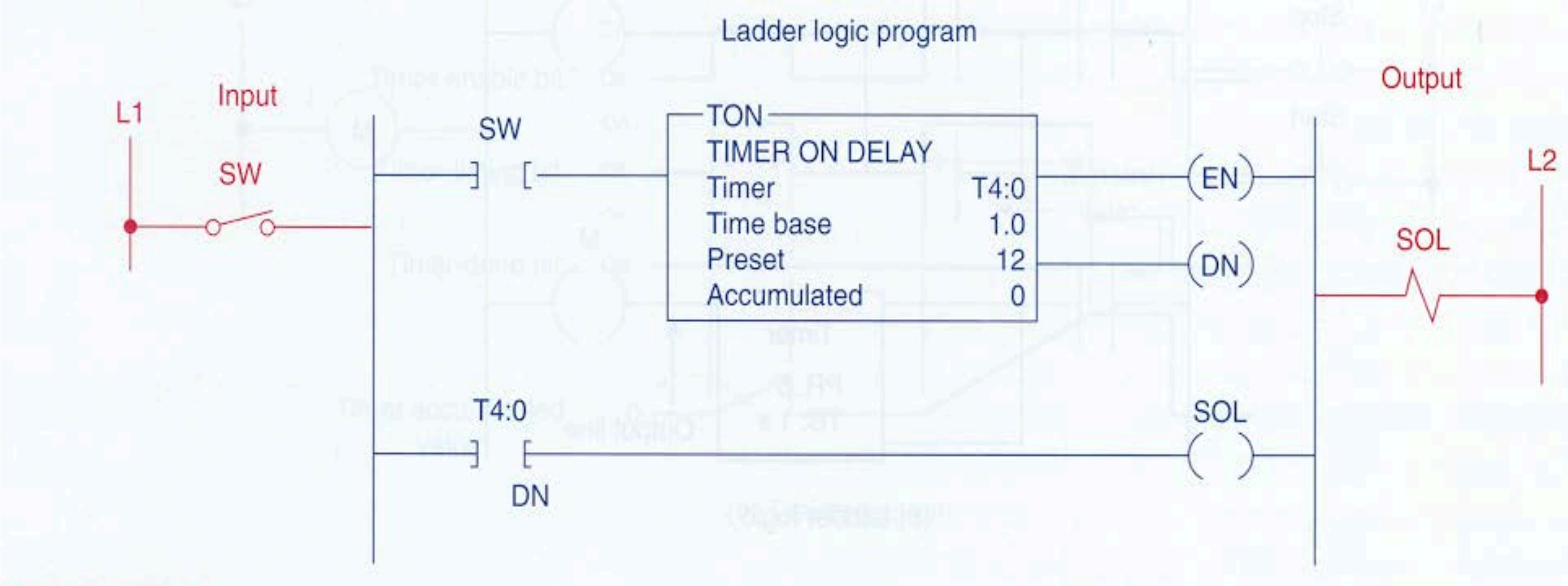


Fig. 7-14

Solenoid valve timed closed.

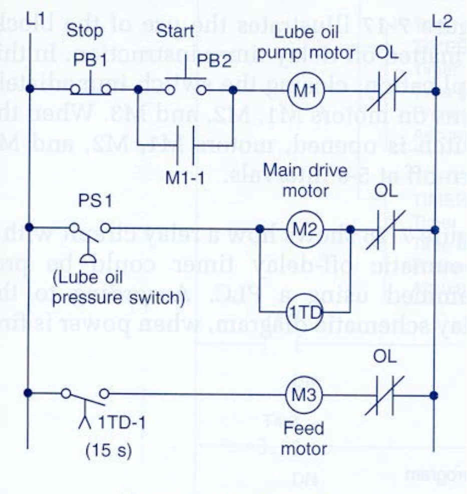
Ladder diagram

Example of *timer on-delay*

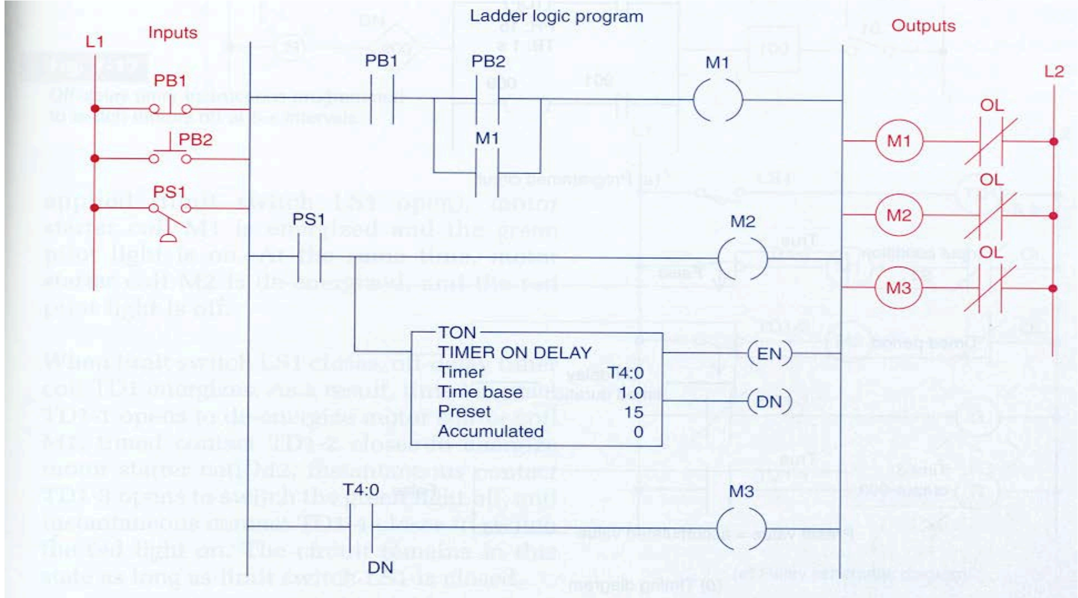
- If PB2 is activated, powers on the oil pumping motor.

- When the pressure augments, PS1 detects the increase and activates the main motor.

- 15 segunds later the main drive motor starts.



(*a*) Relay ladder schematic diagram



(*b*) Ladder logic

Fig. 7-15
Automatic sequential control system.

Ladder diagram

Example of *timer* programmed as *off-delay*

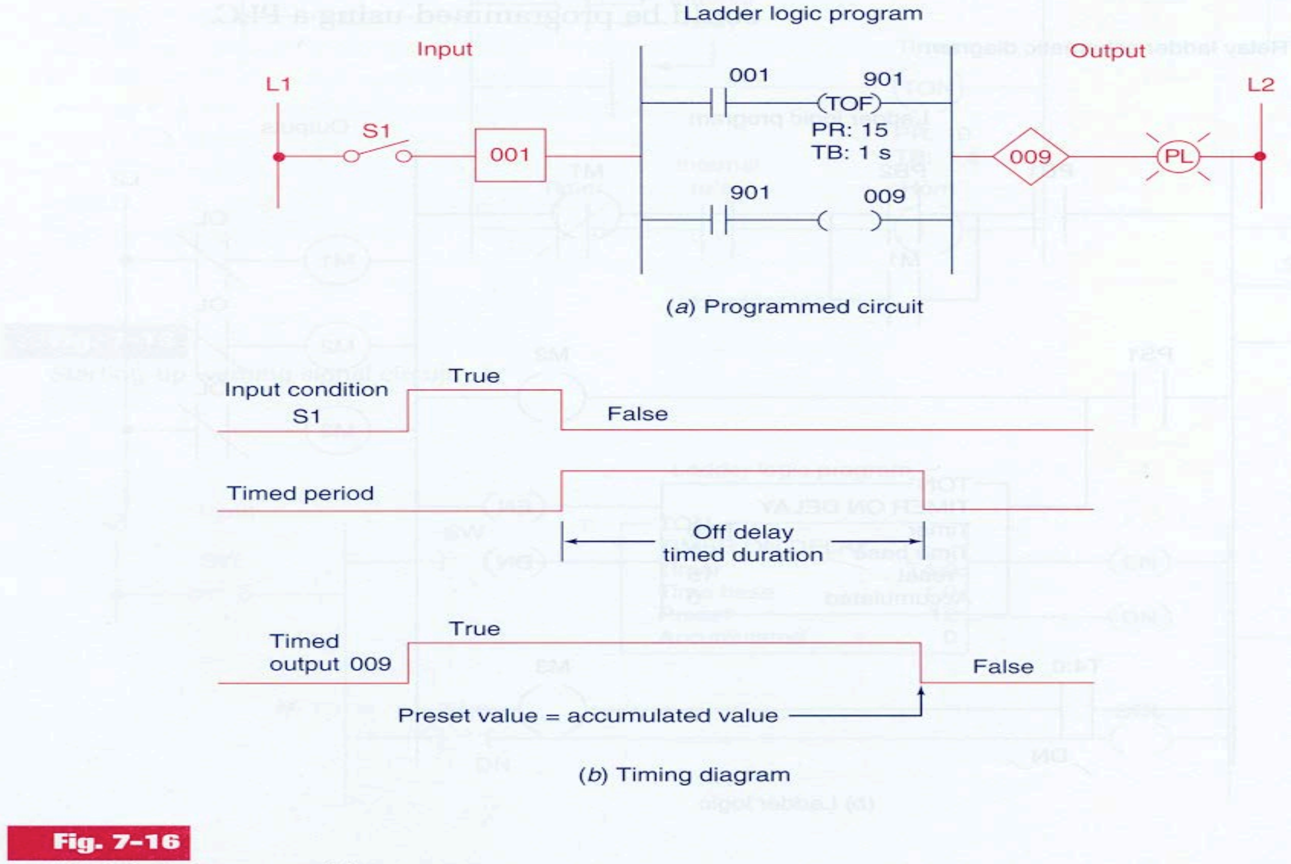


Fig. 7-16

Off-delay programmed timer.

Ladder diagram

Example of *timer* programmed as *off-delay*

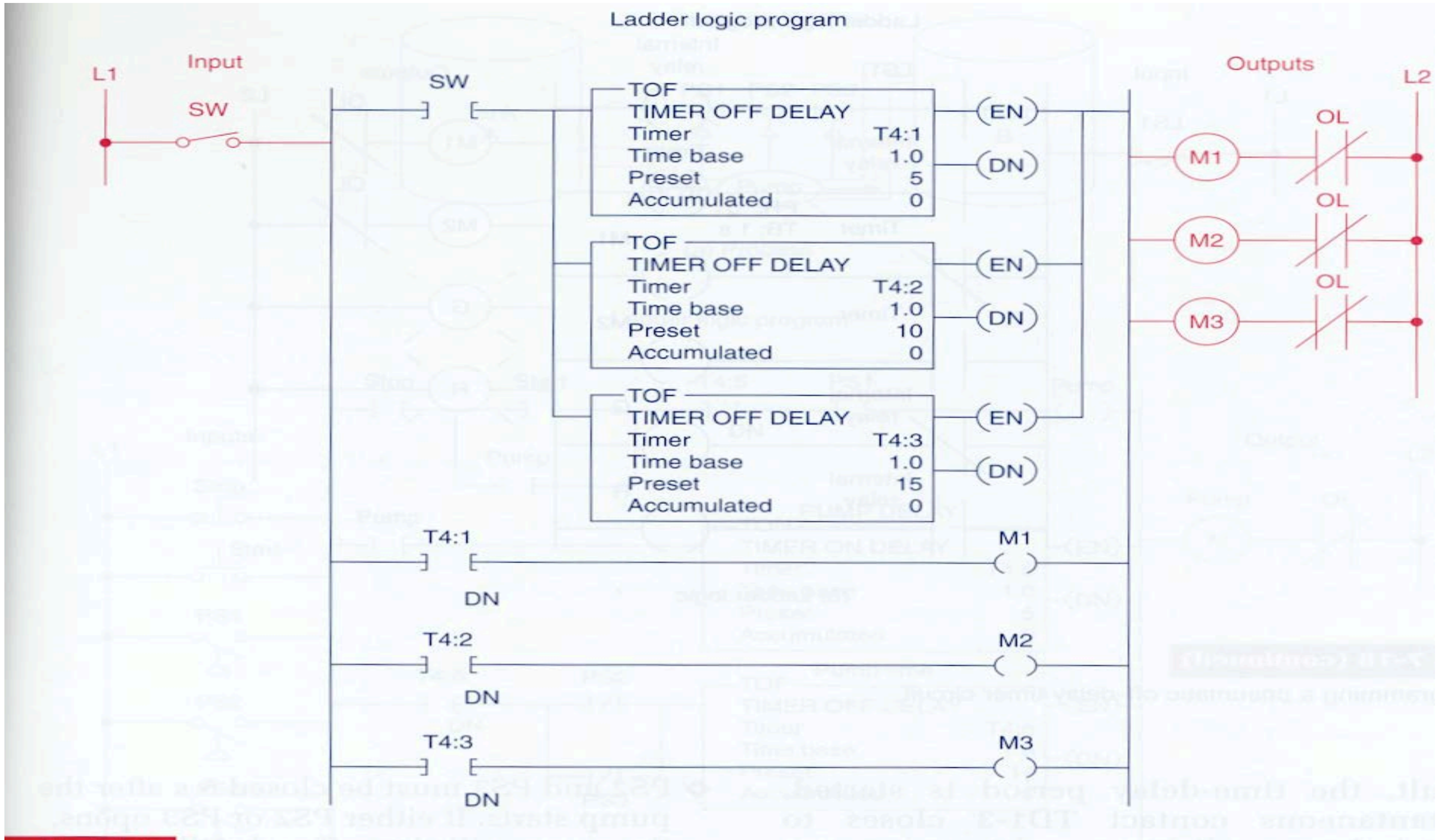
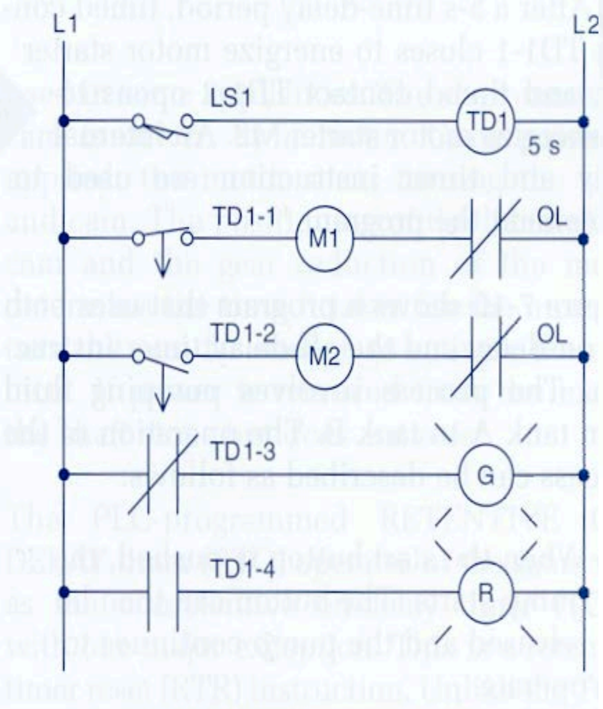


Fig. 7-17
Off-delay timer instructions programmed to switch motors off at 5-s intervals.

Ladder diagram

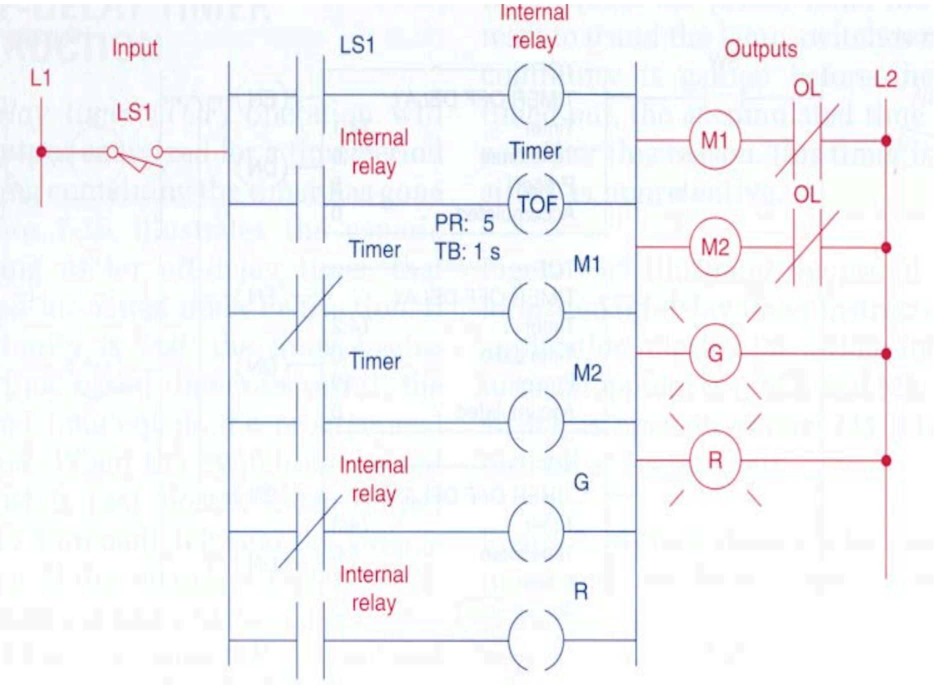
Example of *timer* programmed as *off-delay*



(*a*) Relay schematic diagram

Fig. 7-18

Programming a pneumatic off-delay timer circuit.



(*b*) Ladder logic

Fig. 7-18 (continued)

Programming a pneumatic off-delay timer circuit.

Ladder diagram

Example of *timers* programmed as *off-delay* and *on-delay*

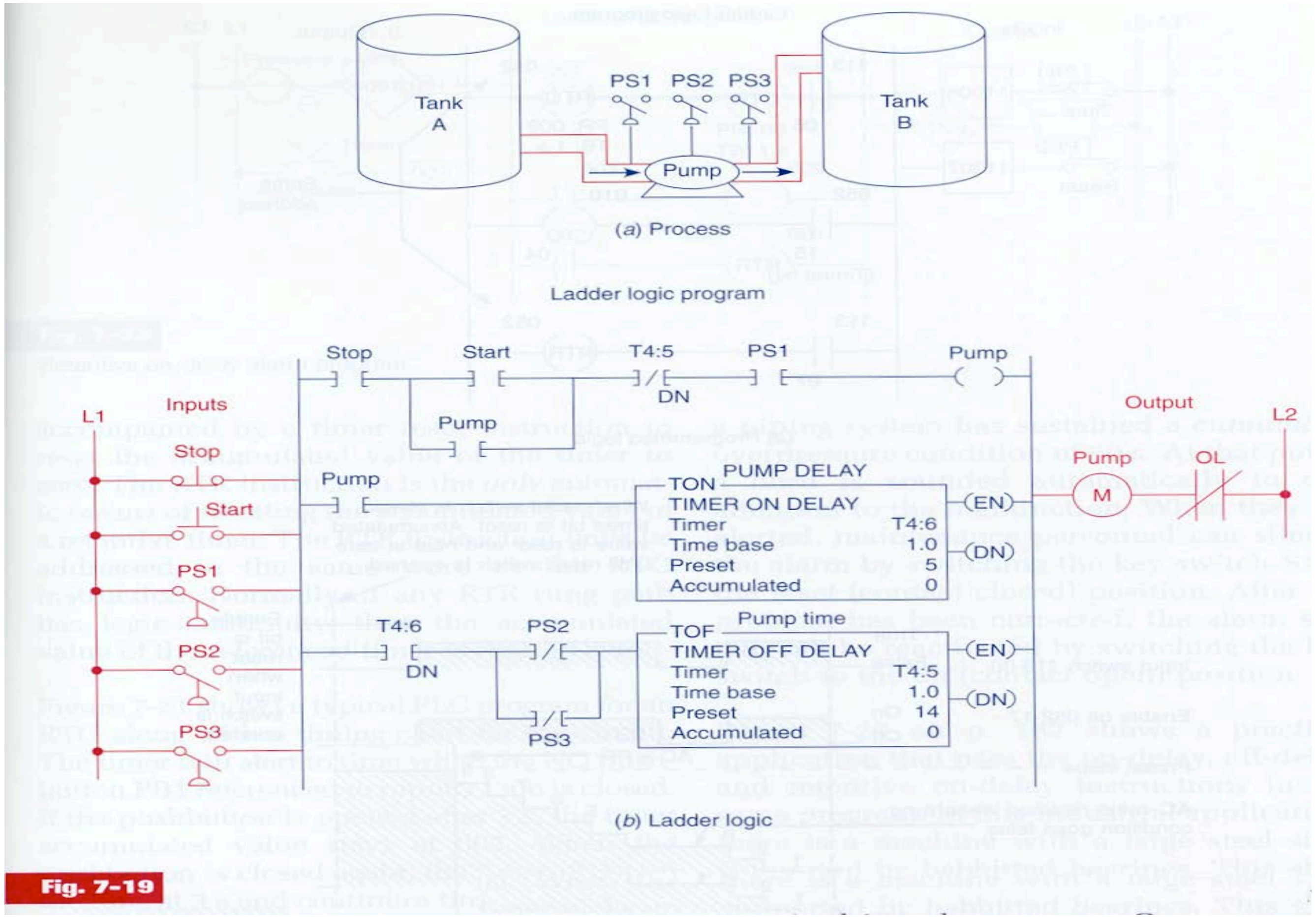
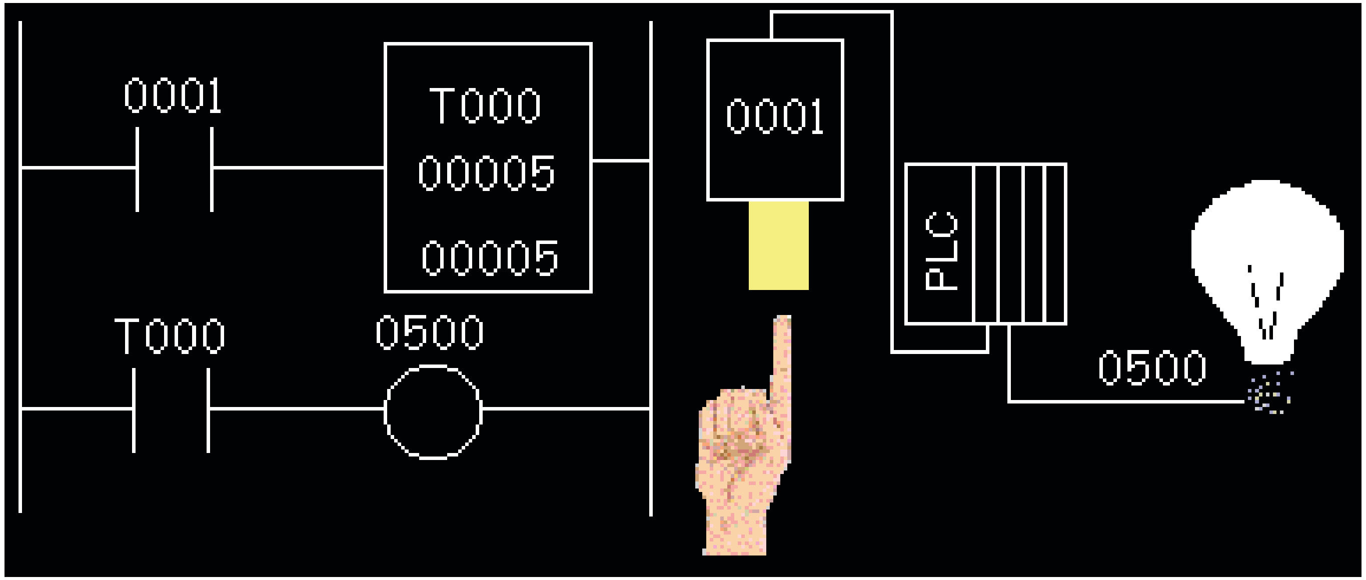


Fig. 7-19

Ladder diagram

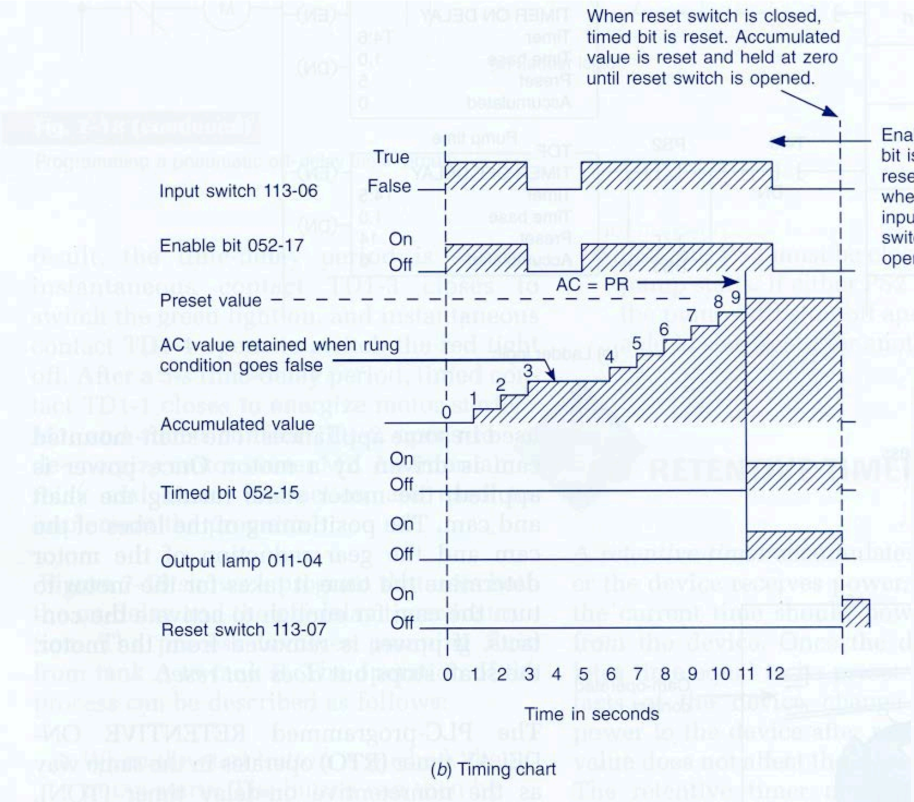
Timers

Example:

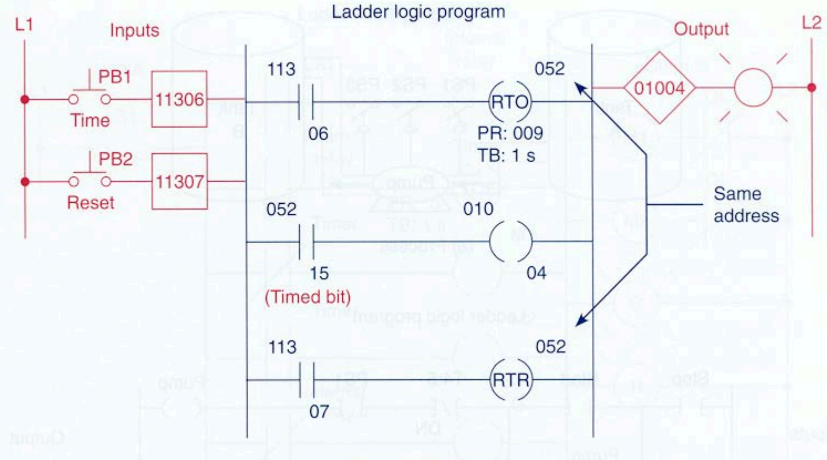


Ladder diagram

Retentive Timers



(*b*) Timing chart



(*a*) Programmed logic

Fig. 7-21

Retentive on-delay timer program and timing chart.

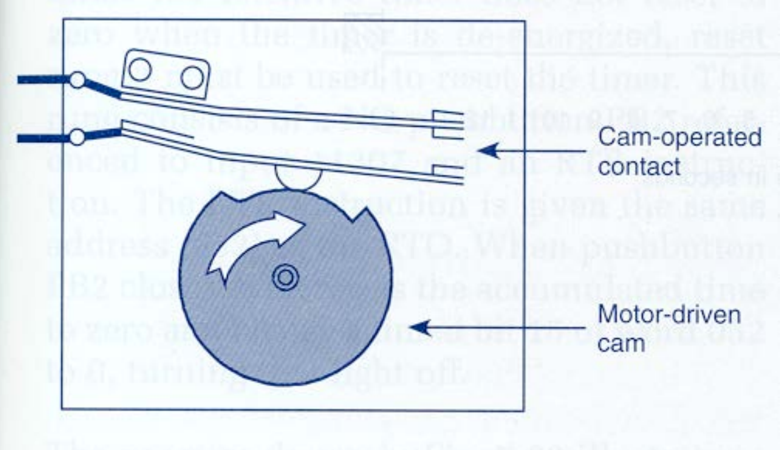


Fig. 7-20

Electromechanical retentive timer.

Ladder diagram

Example of *retentive timers*

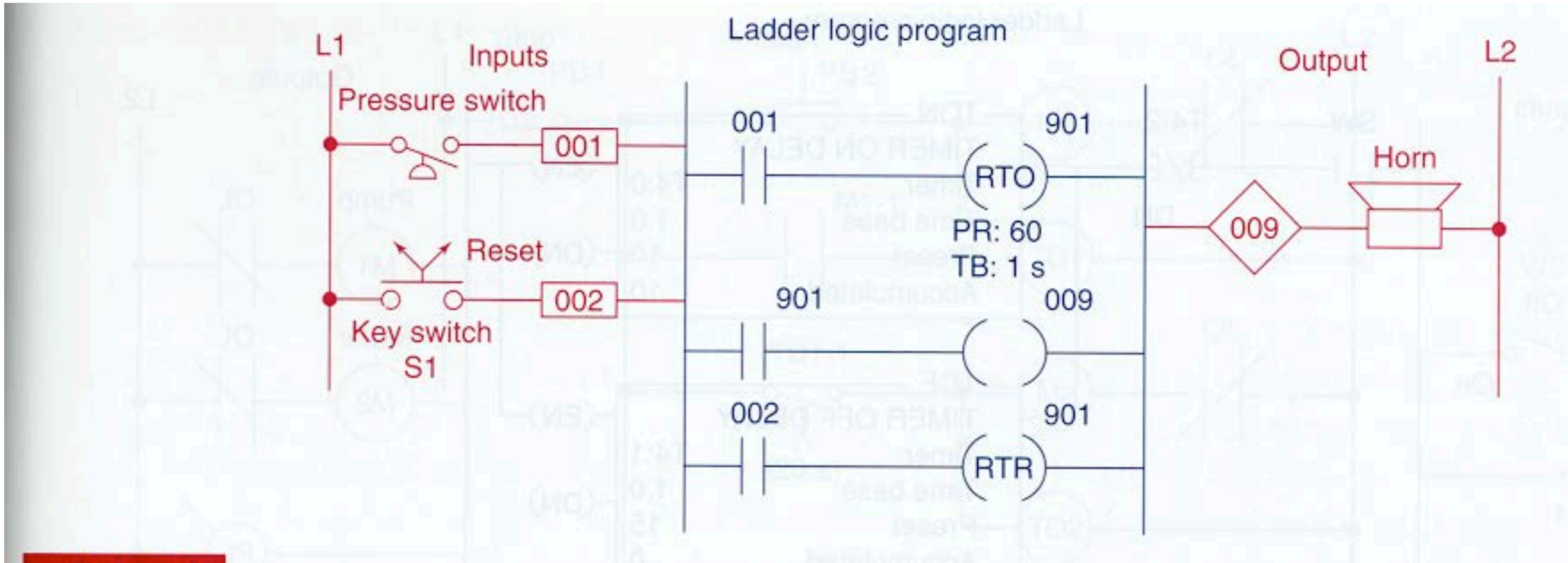


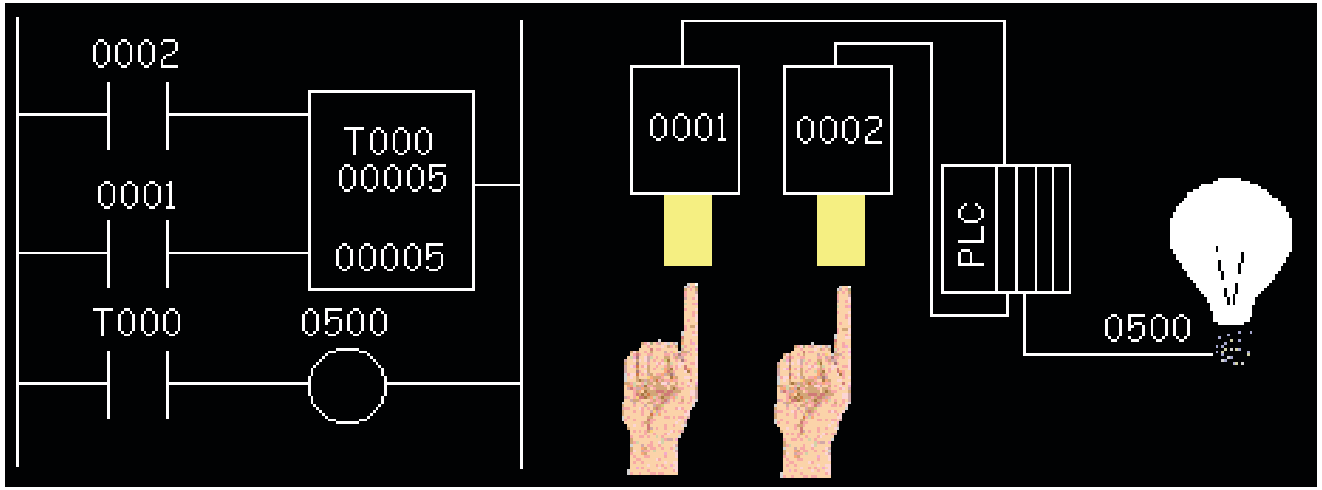
Fig. 7-22

Retentive on-delay alarm program.

Ladder diagram

Timers

Example:
(search on the Schneider PLC or discuss implementation)



Ladder diagram

Exemplo:

- 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.

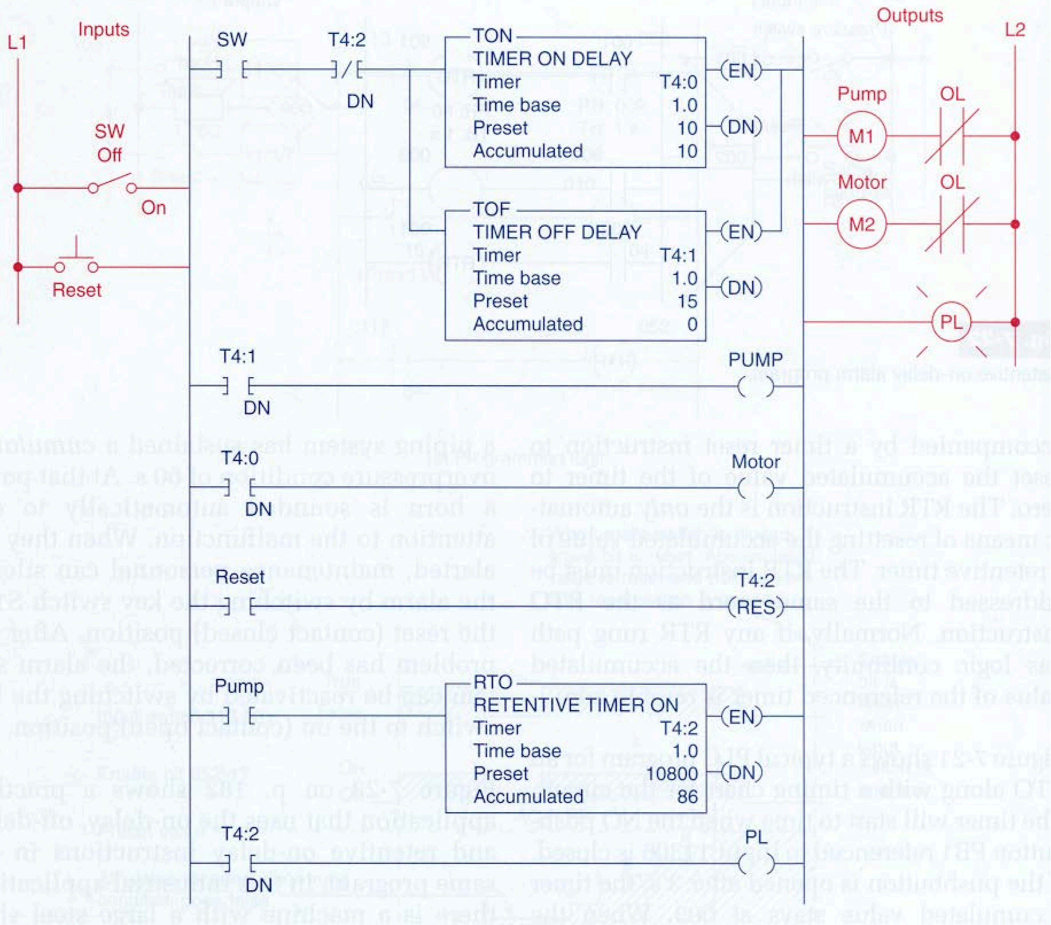
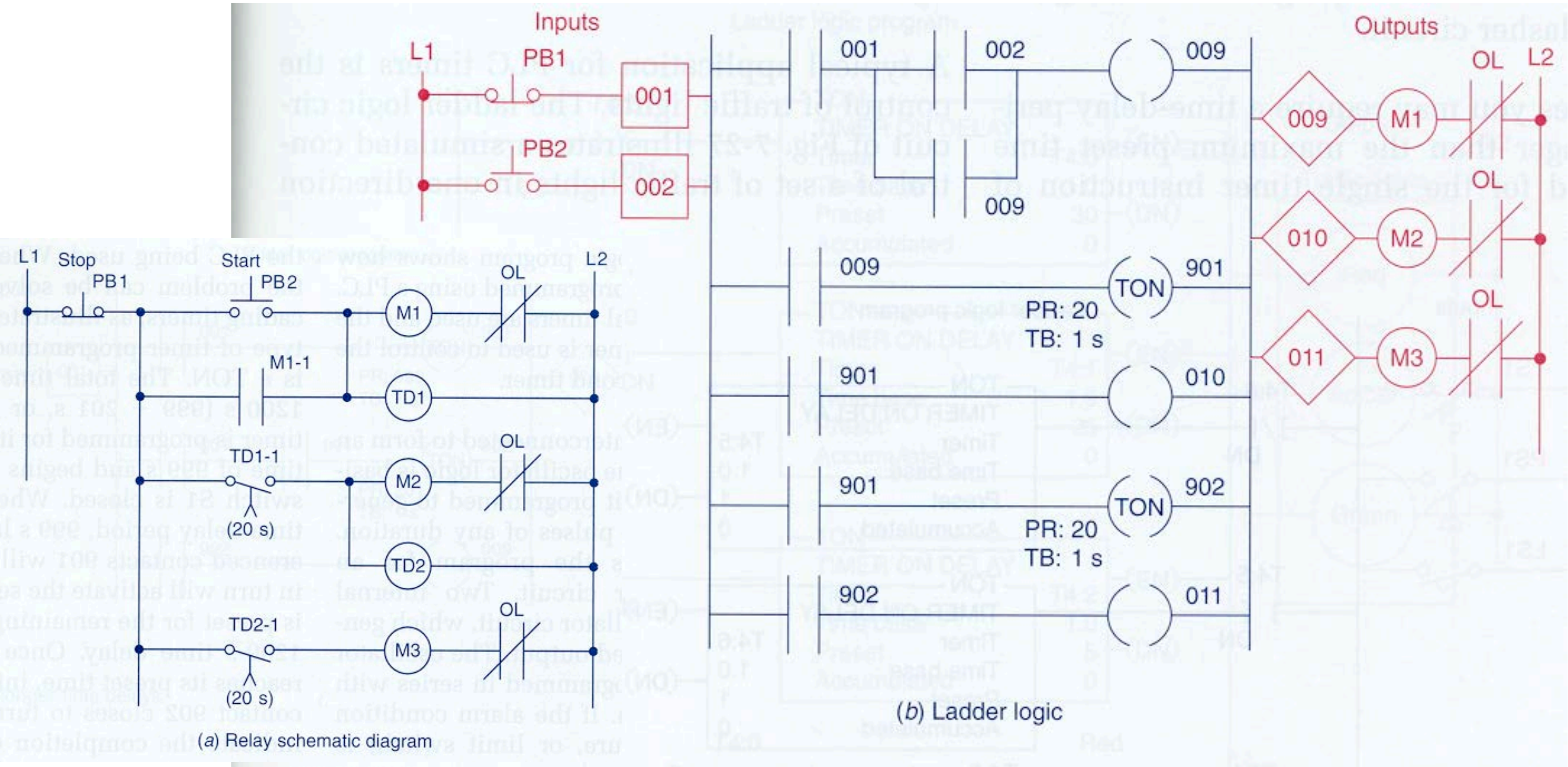


Fig. 7-23 Bearing lubrication program.

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)

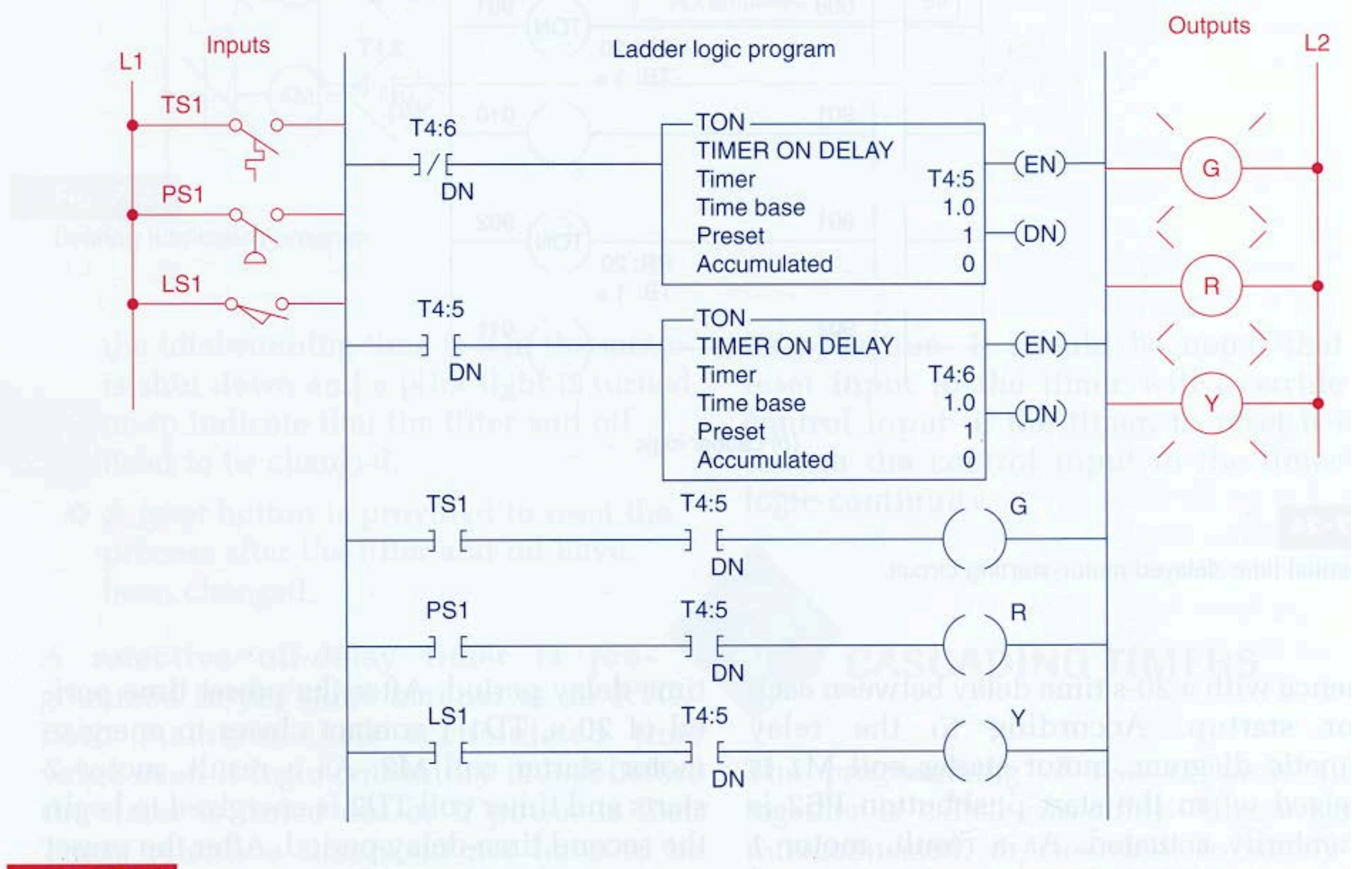


Fig. 7-25

Annunciator flasher program.

Ladder diagram

Timers for very long time intervals

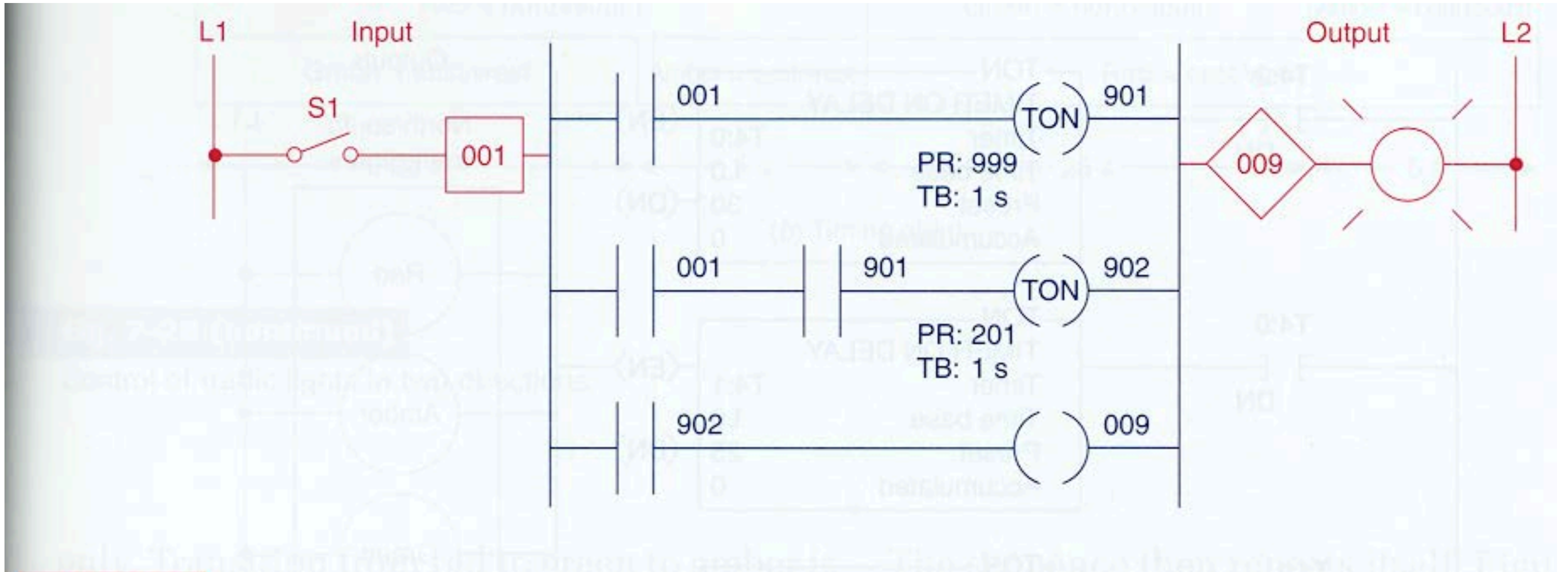


Fig. 7-26
Cascading of timers for longer time delays.

Ladder diagram

Example of a semaphore

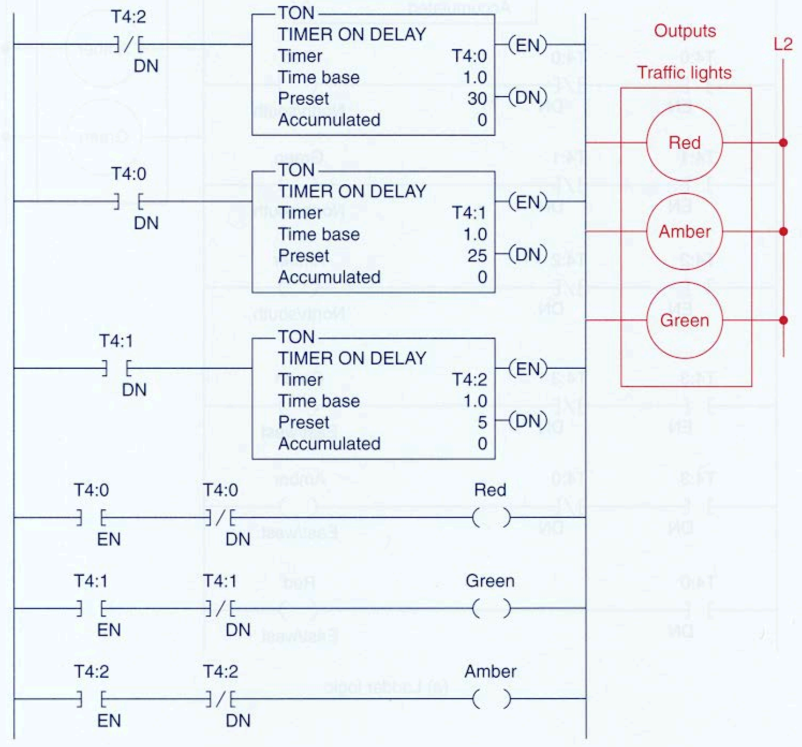
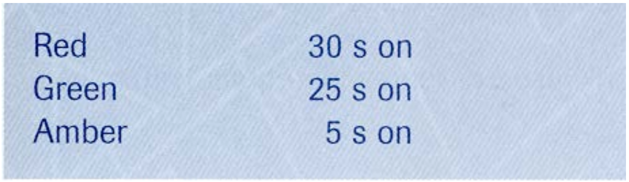
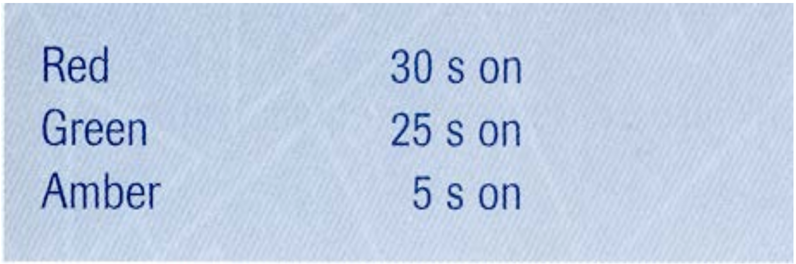


Fig. 7-27
Control of traffic lights in one direction.

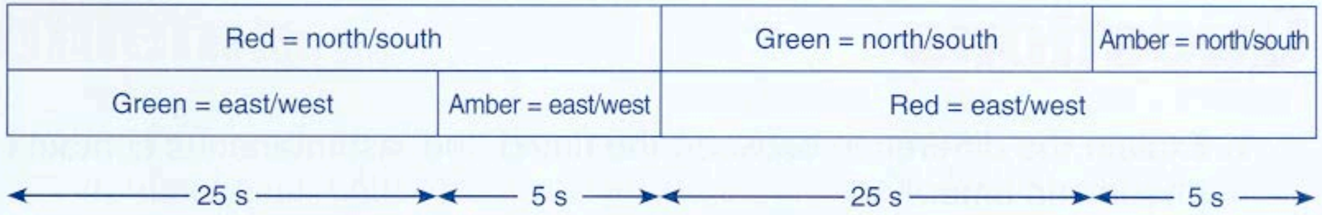


Red	30 s on
Green	25 s on
Amber	5 s on

Example of a semaphore in both directions



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



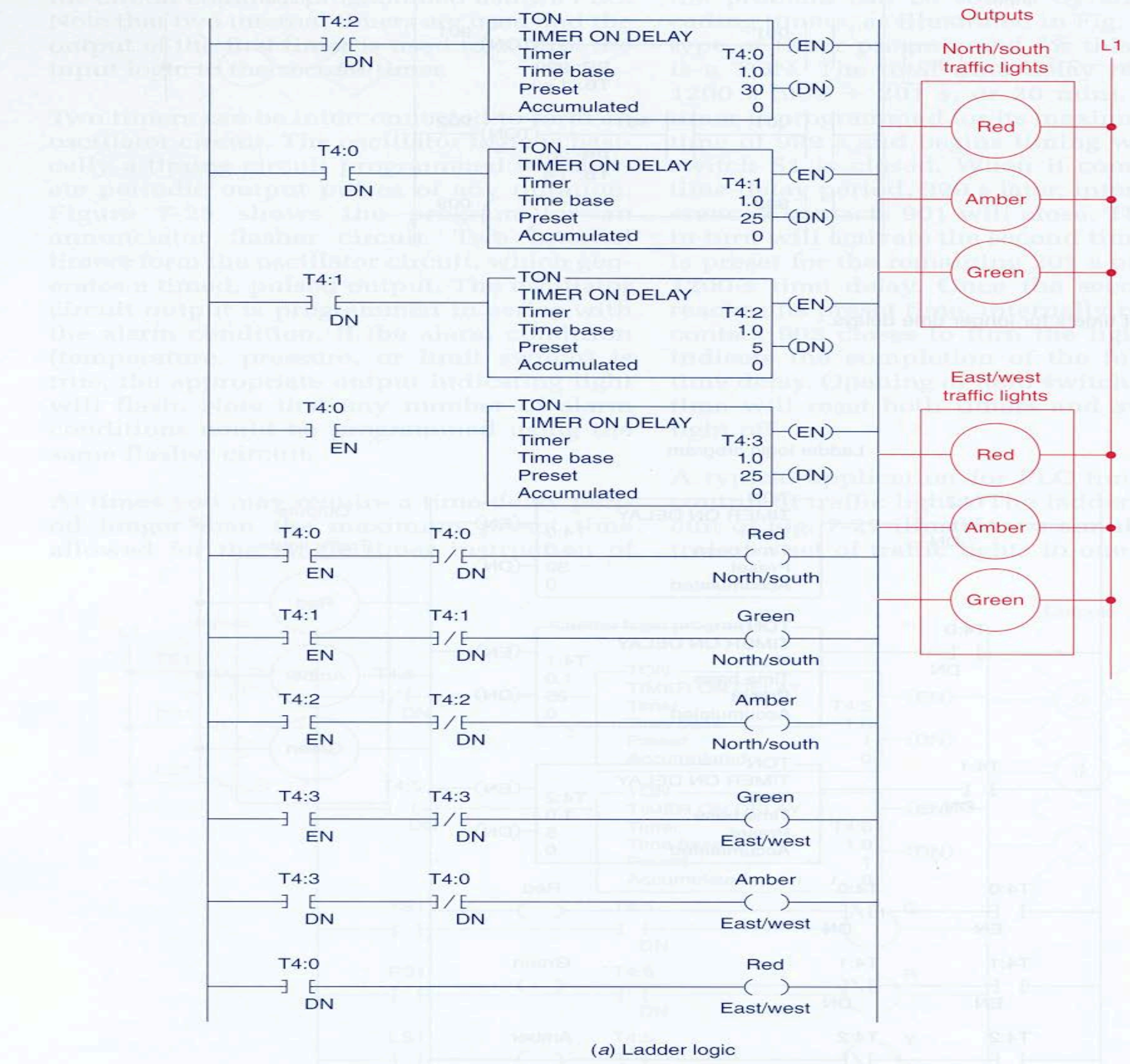
Red = north/south		Green = north/south	Amber = north/south
Green = east/west	Amber = east/west	Red = east/west	

(b) Timing chart

Fig. 7-28 (continued)

Control of traffic lights in two directions.

Example of a semaphore in both directions

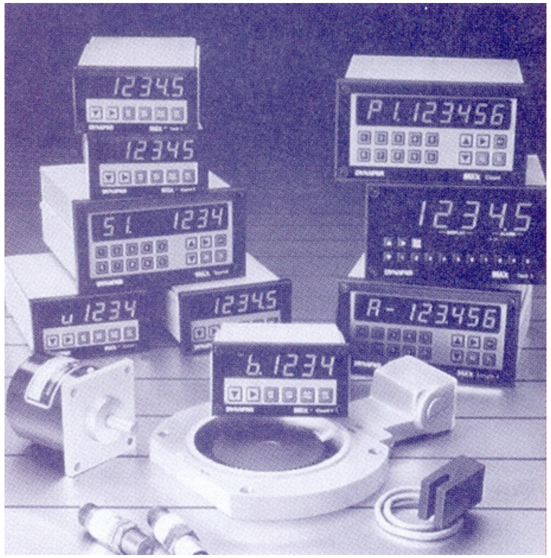


(a) Ladder logic

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

Ladder diagram

Counters



Some applications...

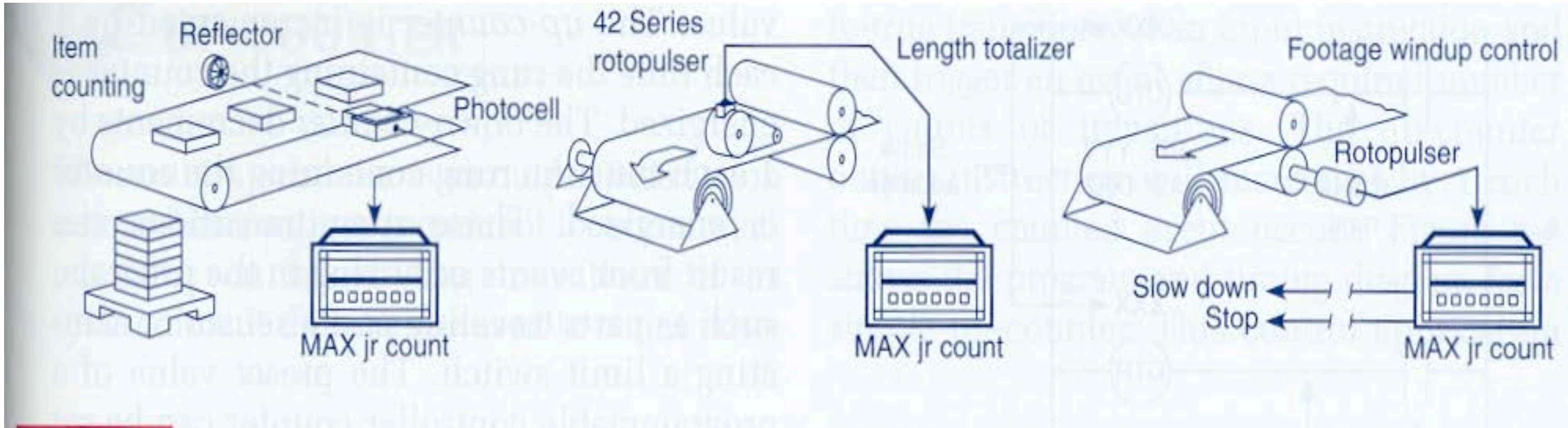
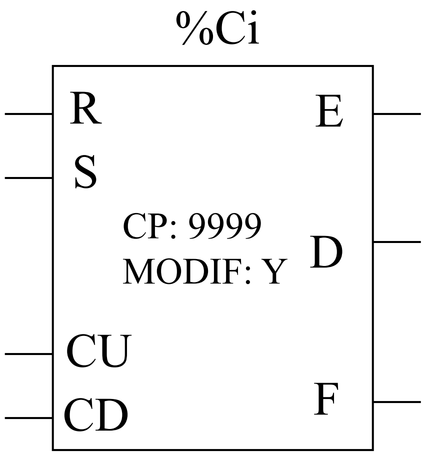


Fig. 8-3

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

Ladder diagram

Counters

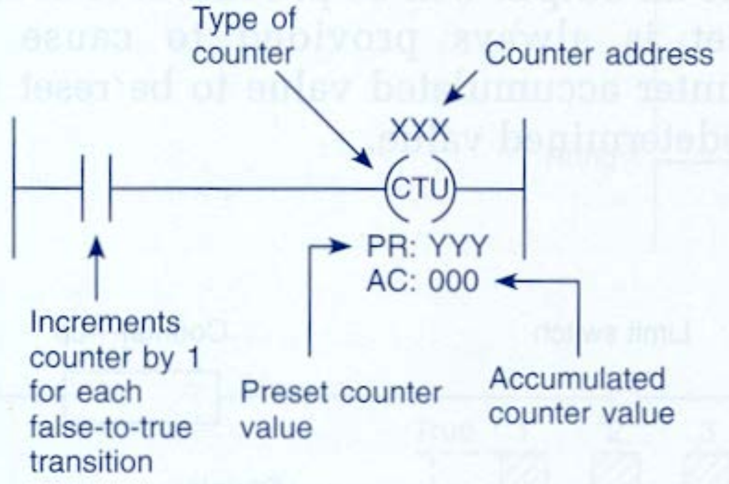


Characteristics:

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

Ladder diagram

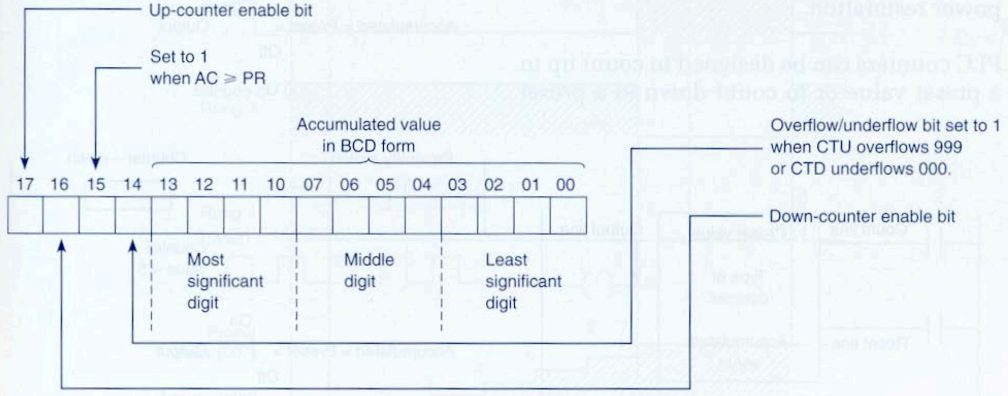
Implementation of Counters in the PLC-5 of *Allen-Bradley*:


Type of counter
Counter address
XXX
CTU
PR: YYY
AC: 000
Increments counter by 1 for each false-to-true transition
Preset counter value
Accumulated counter value

(*a*) Generic instruction

Representation

Internal Structure



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

Fig. 8-4

Coil-formatted counter instruction.

Ladder diagram

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

Alternative representations

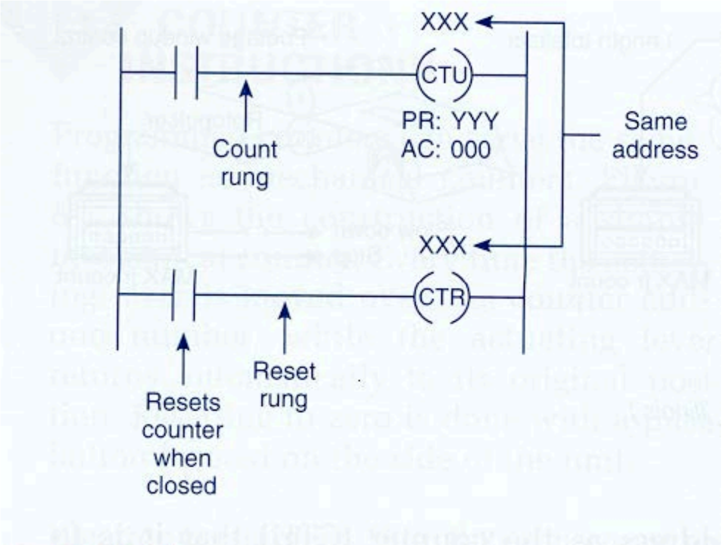


Fig. 8-5

Coil-formatted counter and reset instructions.

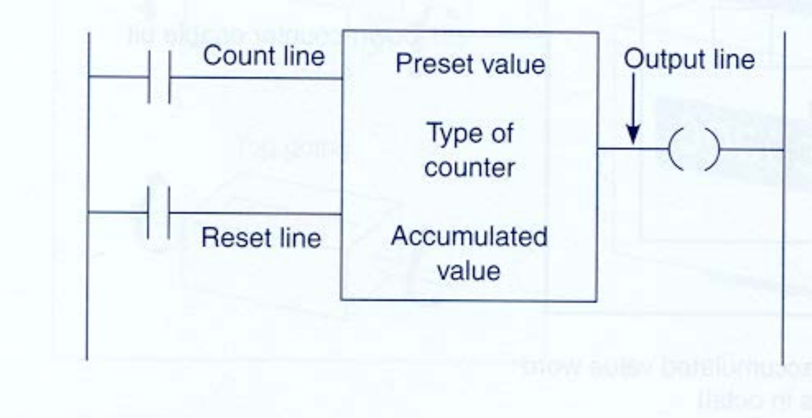
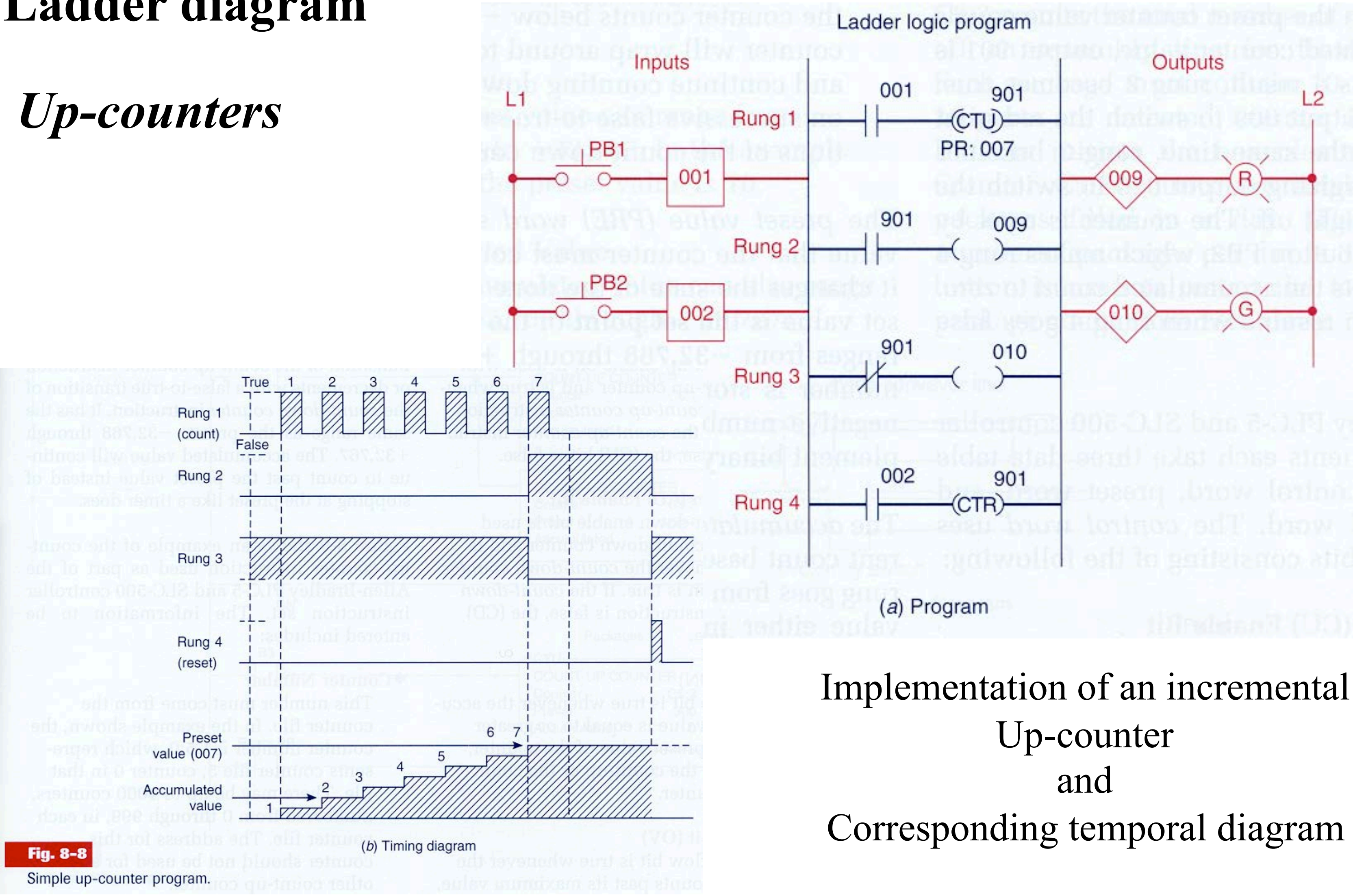


Fig. 8-6

Block-formatted counter instruction.

Ladder diagram

Up-counters



(*a*) Program

(*b*) Timing diagram

Fig. 8-8 Simple up-counter program.

Implementation of an incremental
Up-counter
and
Corresponding temporal diagram

Ladder diagram

Example:

Counting parts

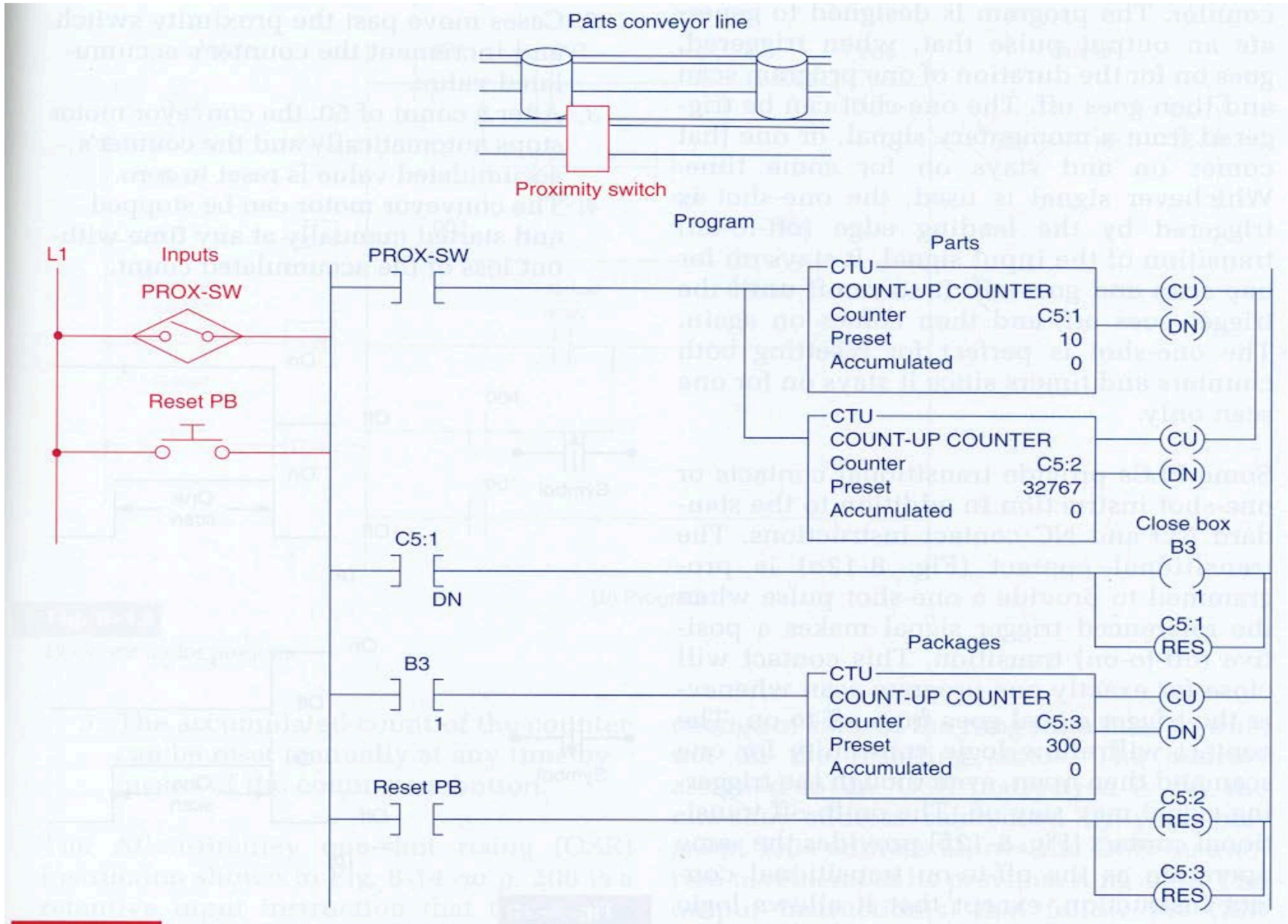
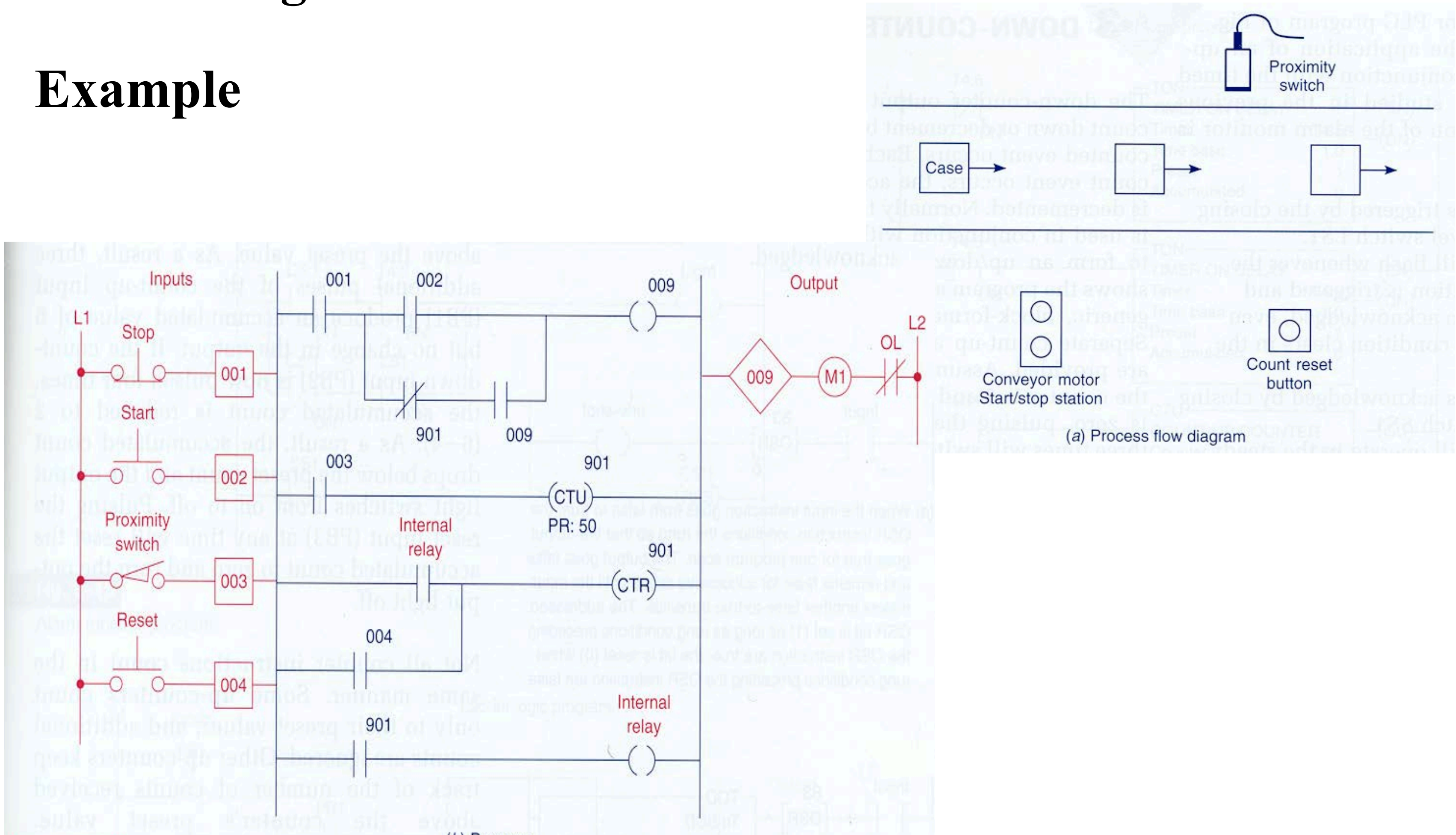


Fig. 8-10 Parts counting program.

Ladder diagram

Example



(a) Process flow diagram

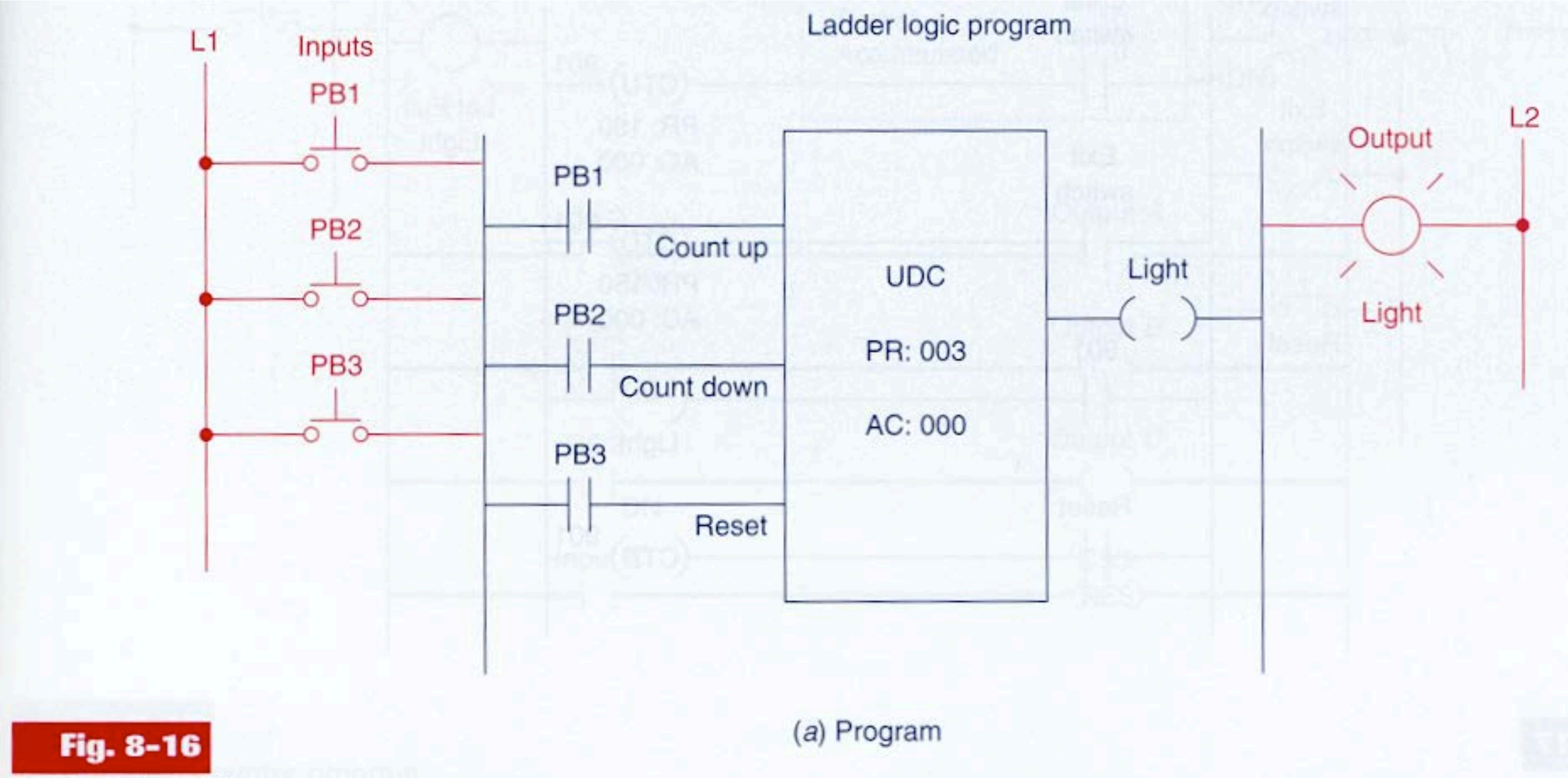
(b) Program

Fig. 8-13

Conveyor motor program.

Ladder diagram

Up/down-counters



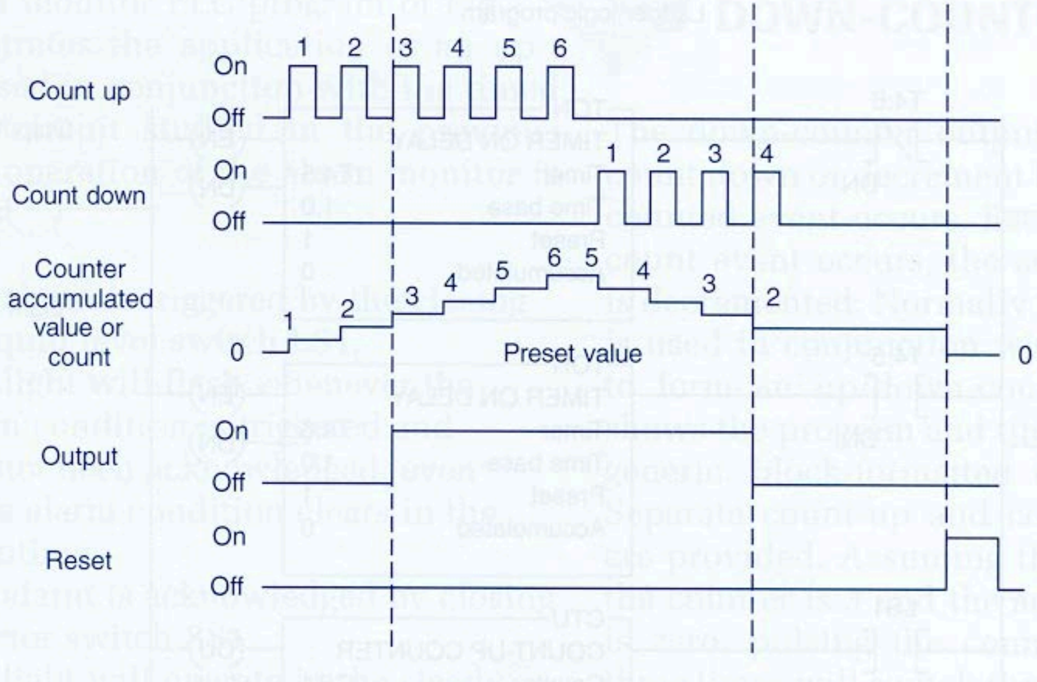
(*a*) Program

Fig. 8-16

Generic up/down-counter program.

Ladder diagram

Up/down-counters



(*b*) Timing diagram

Fig. 8-16 (continued)

Generic up/down-counter program.

Ladder diagram

Up/down-counters

Example:

Finite

parking

garage

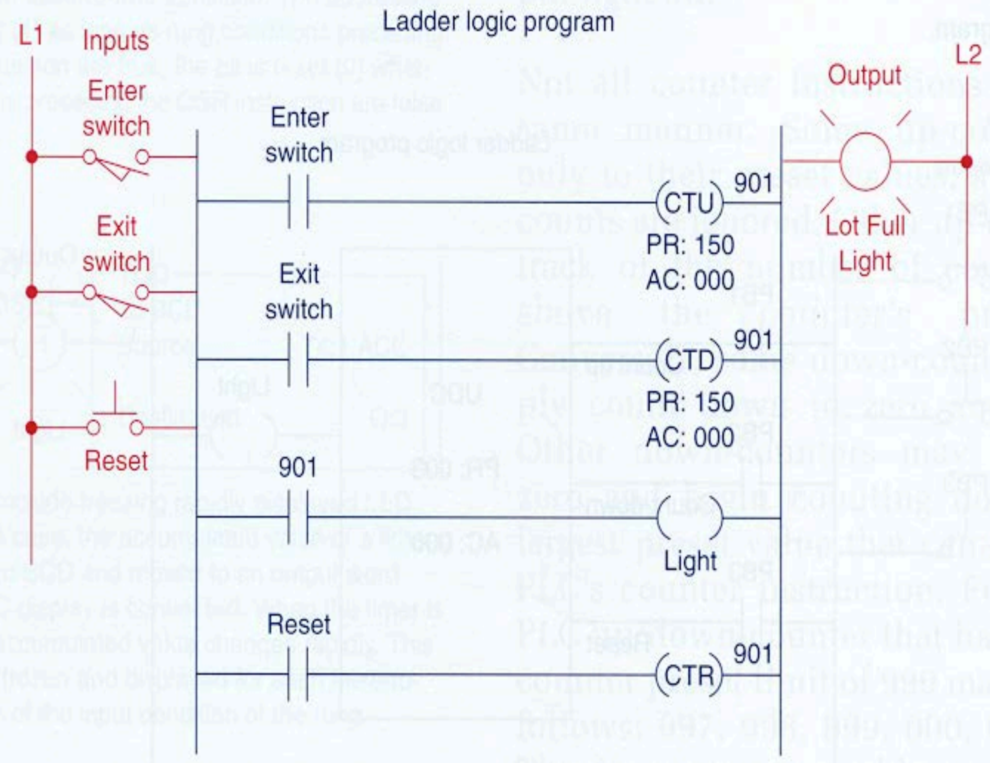


Fig. 8-17

Parking garage counter.

Ladder diagram

Cascaded Counters

Example:

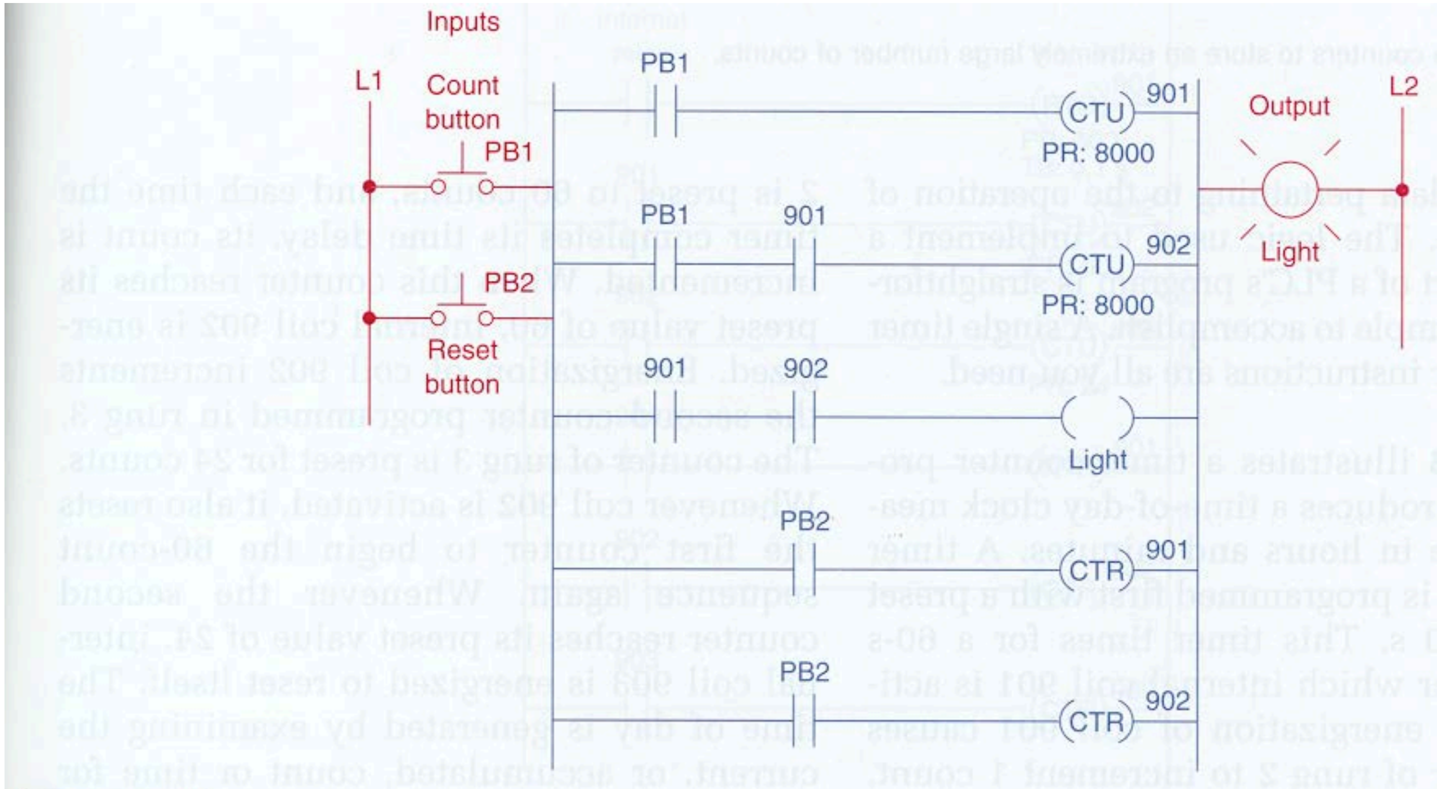


Fig. 8-21

Counting beyond the maximum count.

Ladder diagram

Cascaded Counters

Example:

24 hours clock

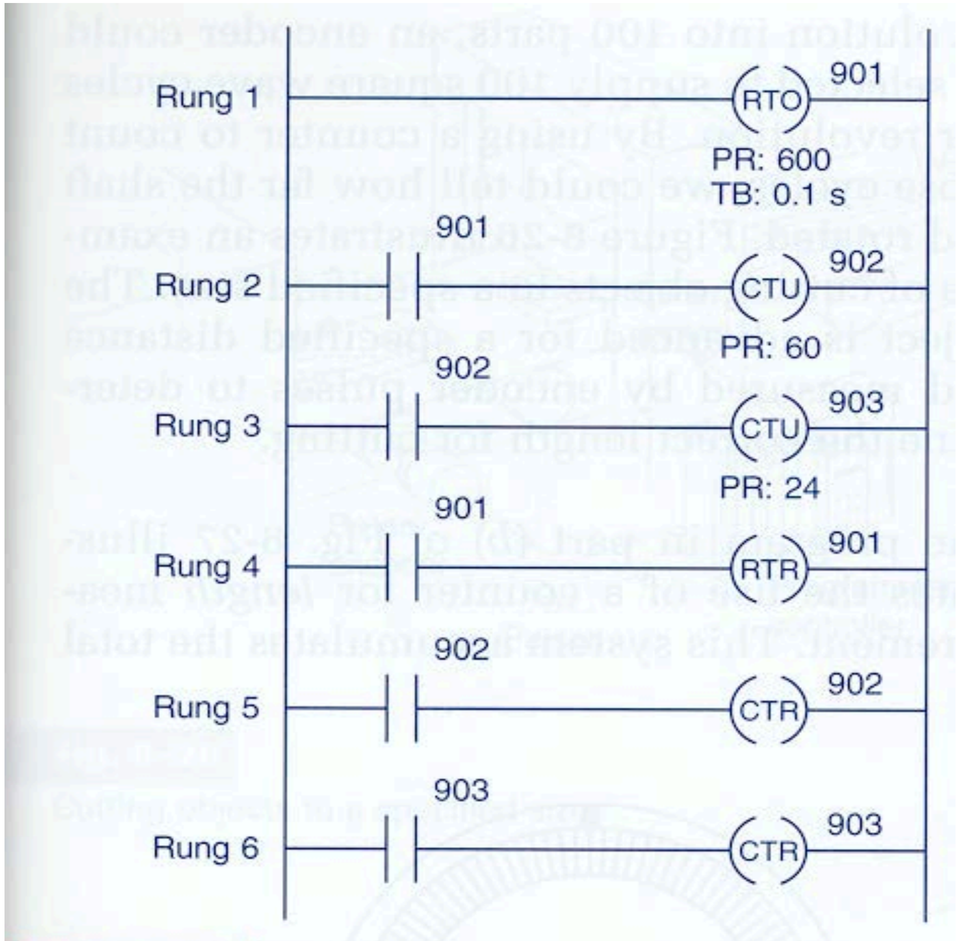


Fig. 8-23
A 24-h clock program.

Ladder diagram

Cascaded Counters

Example:

Memory time of event

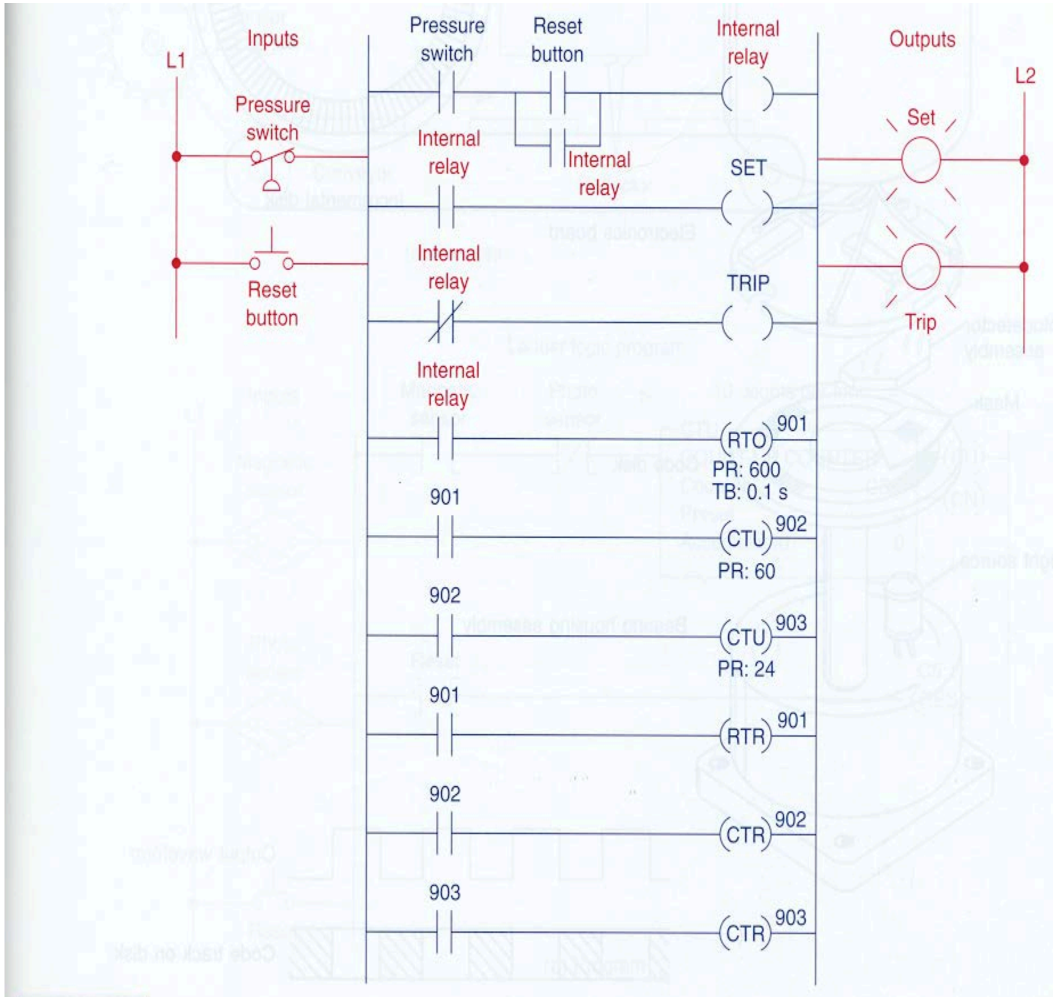


Fig. 8-24

Program for monitoring the time of an event.

Ladder diagram

Incremental *Encoder*

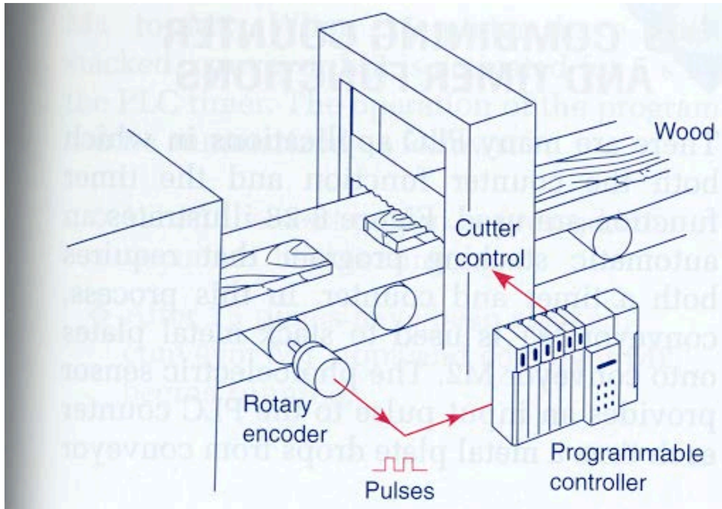


Fig. 8-26 Cutting objects to a specified size.

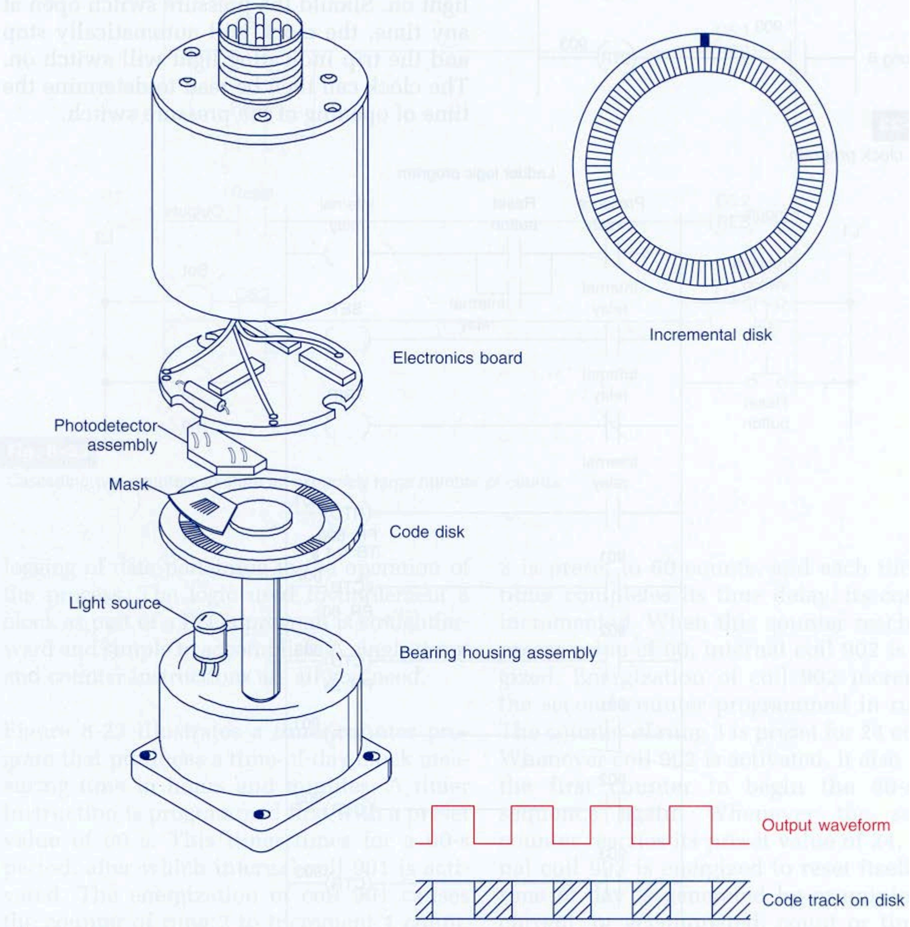


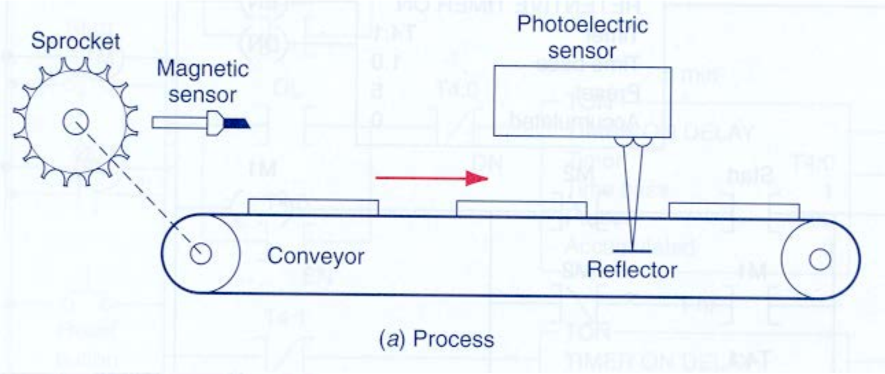
Fig. 8-25 Incremental encoder. *(Courtesy of BEI Motion Systems Company.)*

Ladder diagram

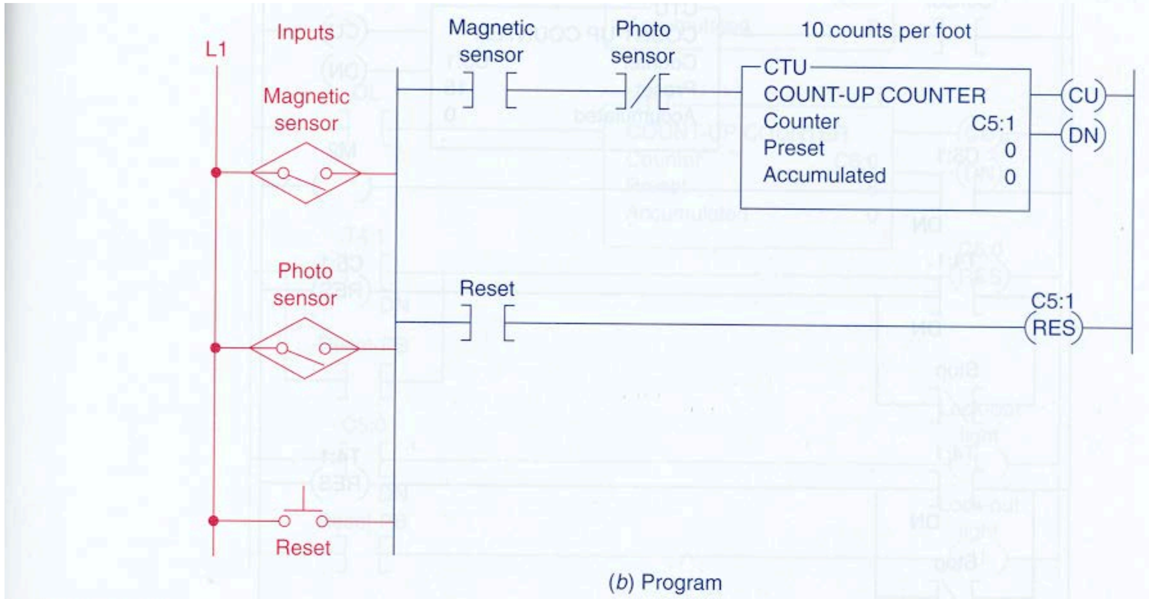
Incremental *Encoder*

Example:

counter as a ”lenght sensor”



(*a*) Process



(*b*) Program

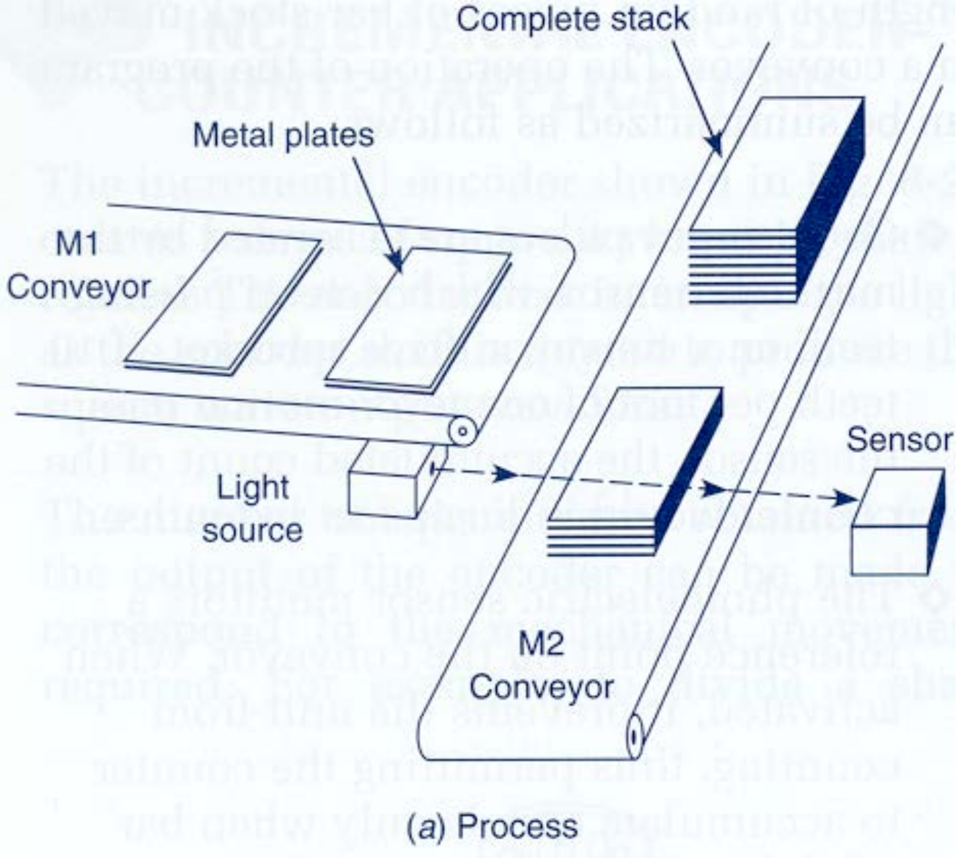
Fig. 8-27 Counter used for length measurement.

Ladder diagram

Example with counters and timers (conts.):

Specs:

- Starts M1 conveyor upon pushing button .
- After 15 plates stops M1 and starts conveyor M2 .
- M2 operates for 5 seconds and then stops.
- Restart sequence.



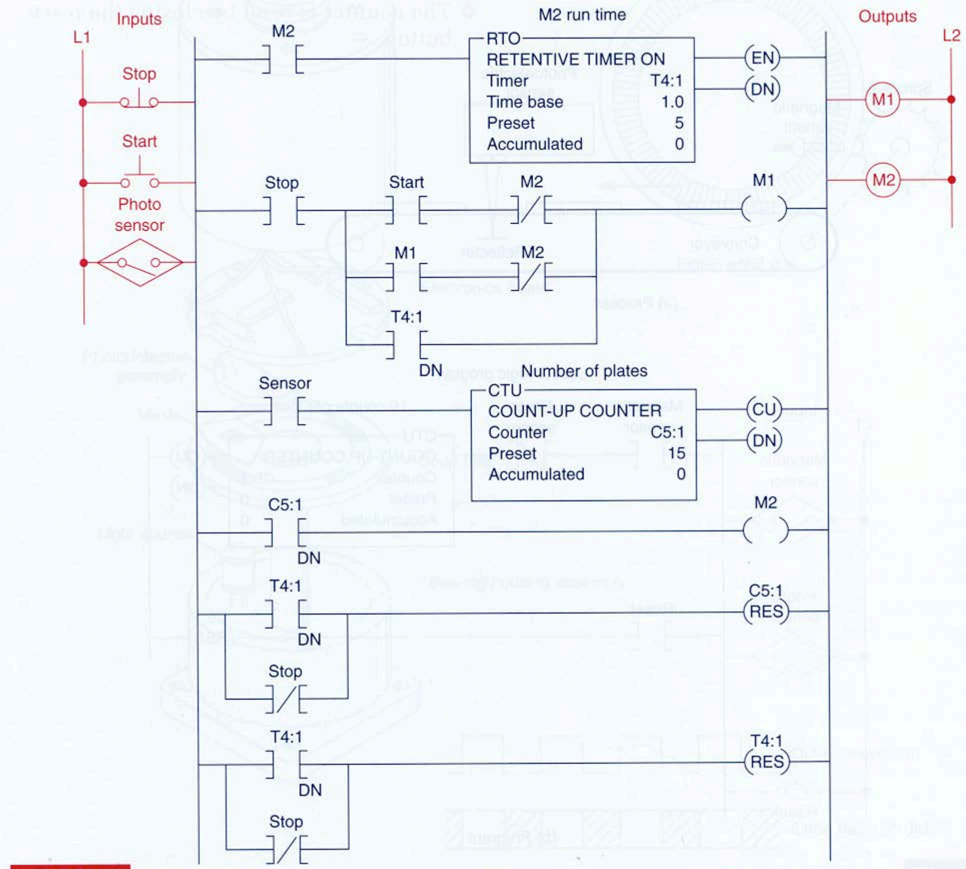
(*a*) Process

Ladder diagram

Example with counters and timers (conts.):

Specs:

- Starts M1 conveyor upon pushing button .
- After 15 plates stops M1 and starts conveyor M2 .
- M2 operates for 5 seconds and then stops.
- Restart sequence.



(*b*) Program

Fig. 8-28 Automatic stacking program.

Ladder diagram

Example with counters and timers (conts.):

Specs:

- Starts M1 conveyor upon pushing button .
- After 15 plates stops M1 and starts conveyor M2 .
- M2 operates for 5 seconds and then stops.
- Restart sequence.

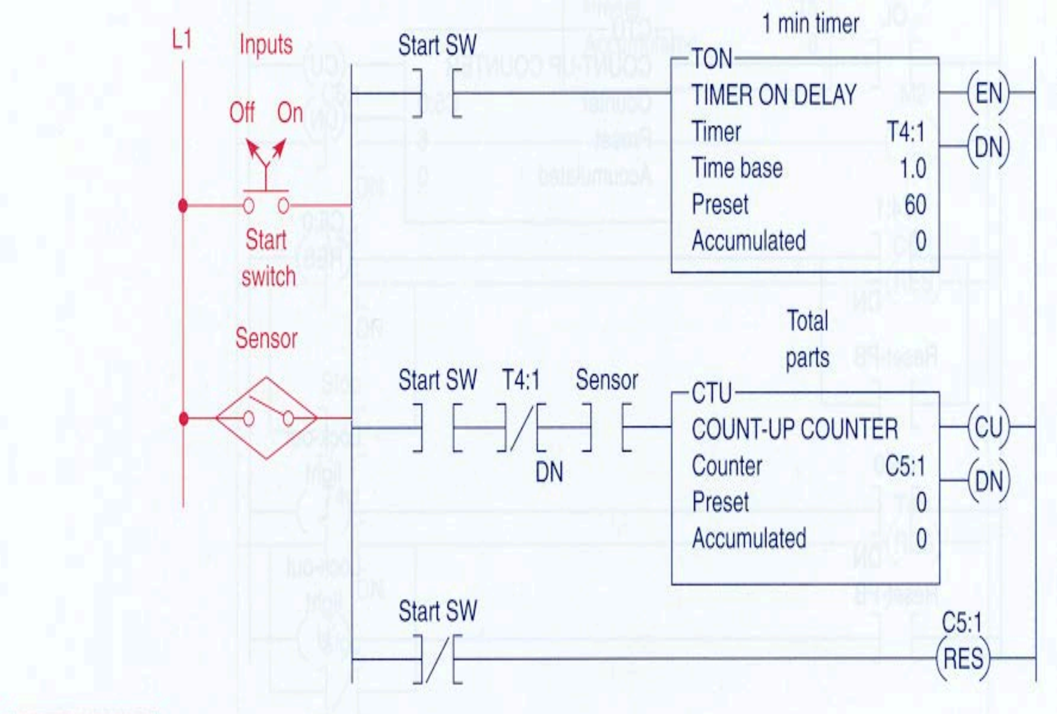


Fig. 8-30
Product flow rate program.

Ladder diagram

Example with counters and timers (conts.):

To use a timer to command a counter, to implement large periods of time.

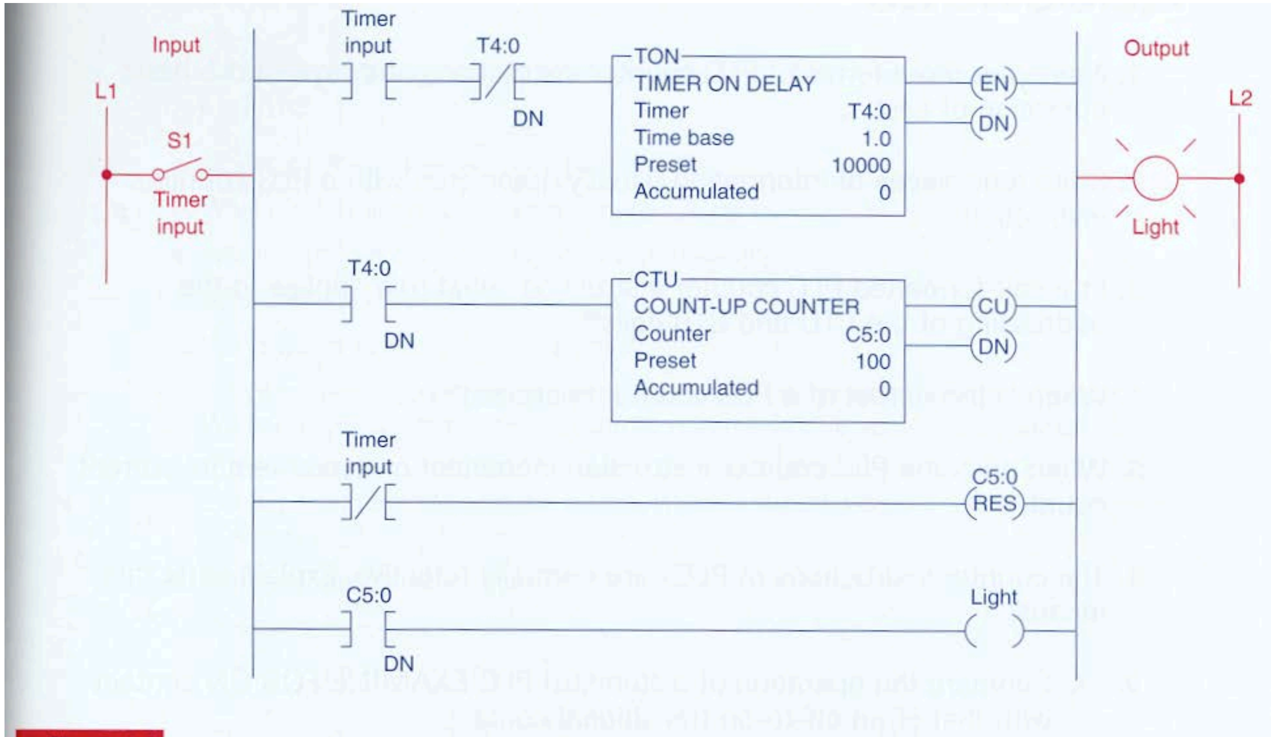
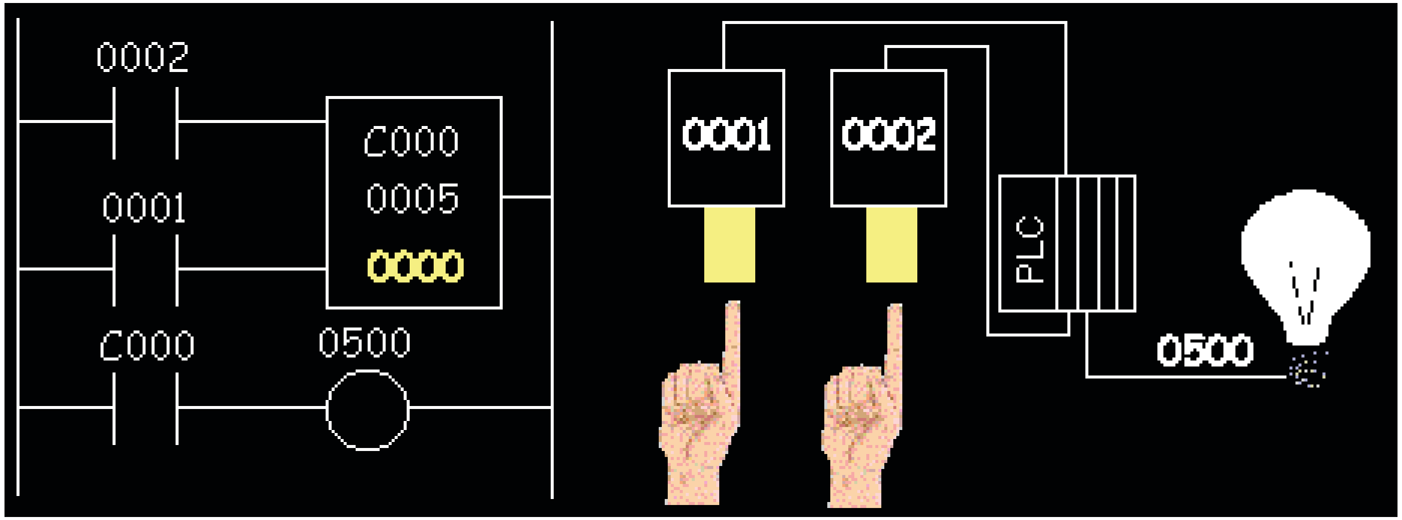


Fig. 8-31
Timer driving a counter to produce an extremely long time-delay period.

Ladder diagram

Counters

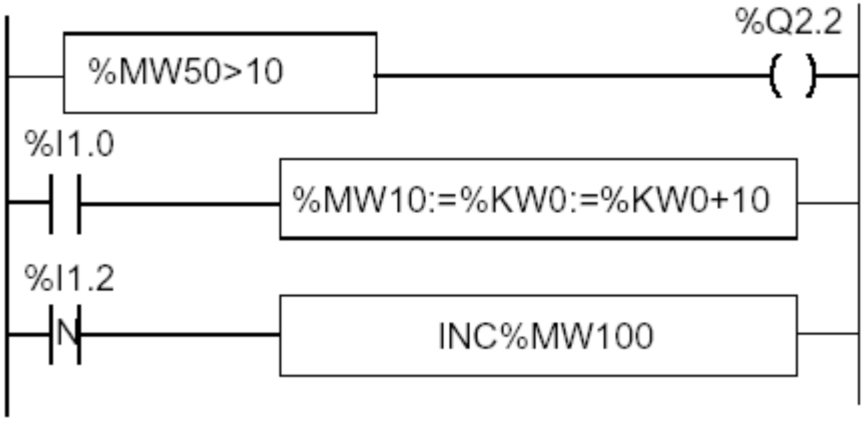
Example:



Ladder diagram

Numerical Processing

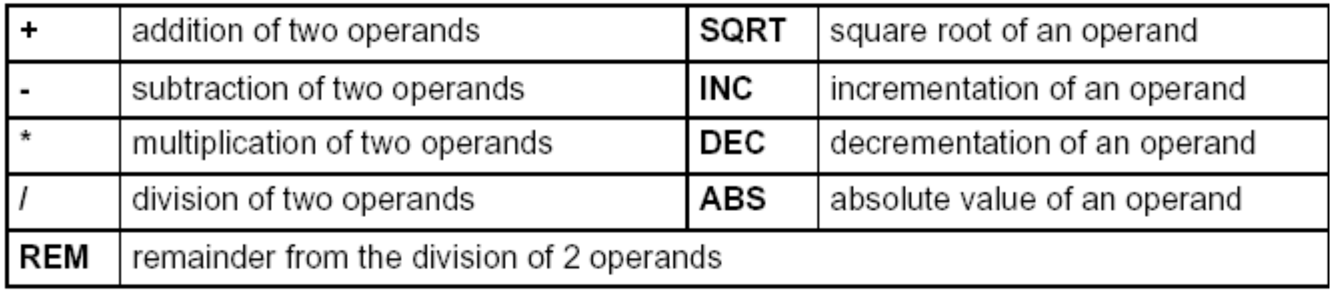
Algebraic and Logic Functions



Ladder diagram

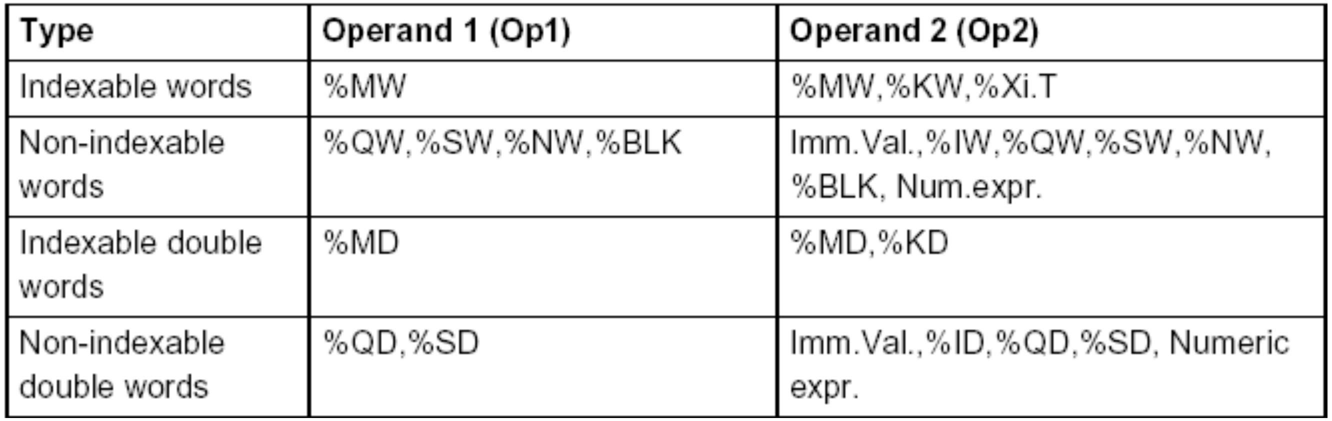
Numerical Processing

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



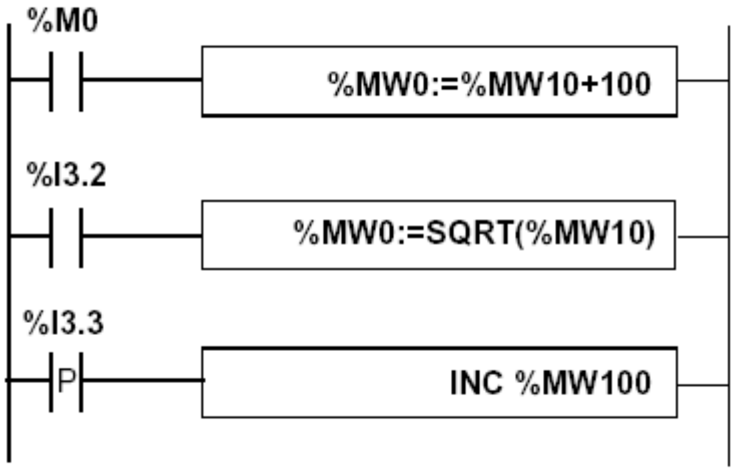
Type	**Operand 1 (Op1)**	**Operand 2 (Op2)**
Indexable words	%MW	%MW,%KW,%Xi.T
Non-indexable words	%QW,%SW,%NW,%BLK	Imm.Val.,%IW,%QW,%SW,%NW, %BLK, Num.expr.
Indexable double words	%MD	%MD,%KD
Non-indexable double words	%QD,%SD	Imm.Val.,%ID,%QD,%SD, Numeric expr.

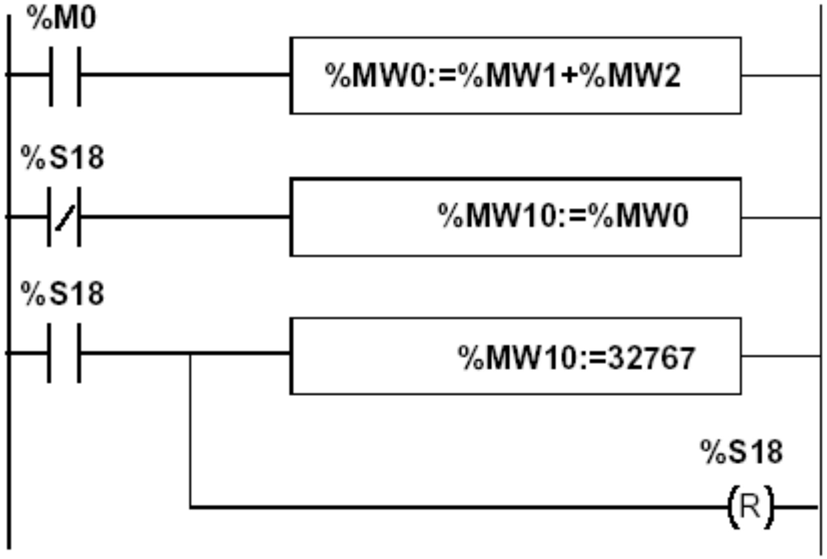
Ladder diagram

Numerical Processing

Example:

Arithmetic functions





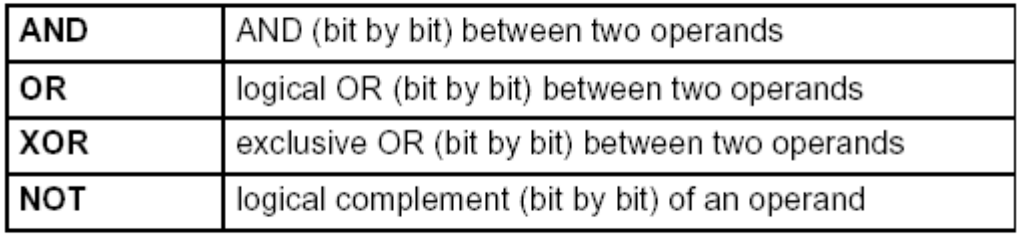
Use of a system variable:

%S18 – flag de overflow

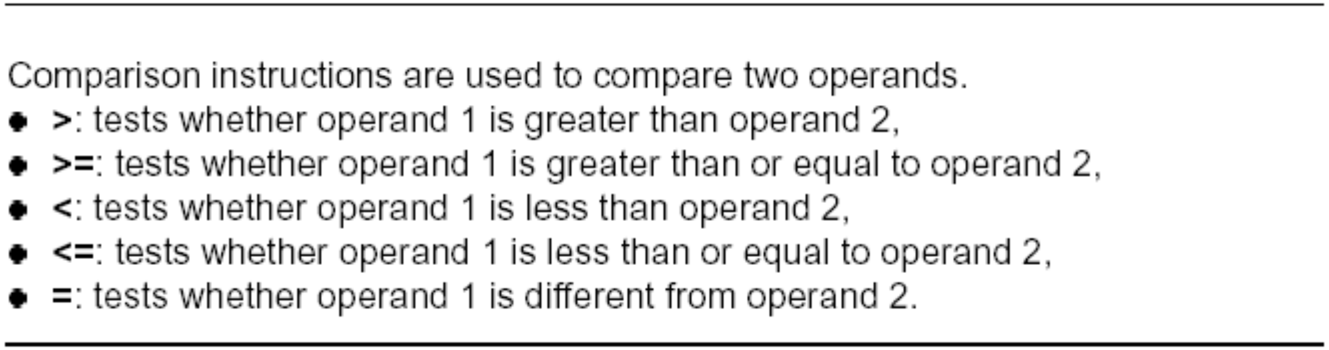
Ladder diagram

Numerical Processing

Logic Functions

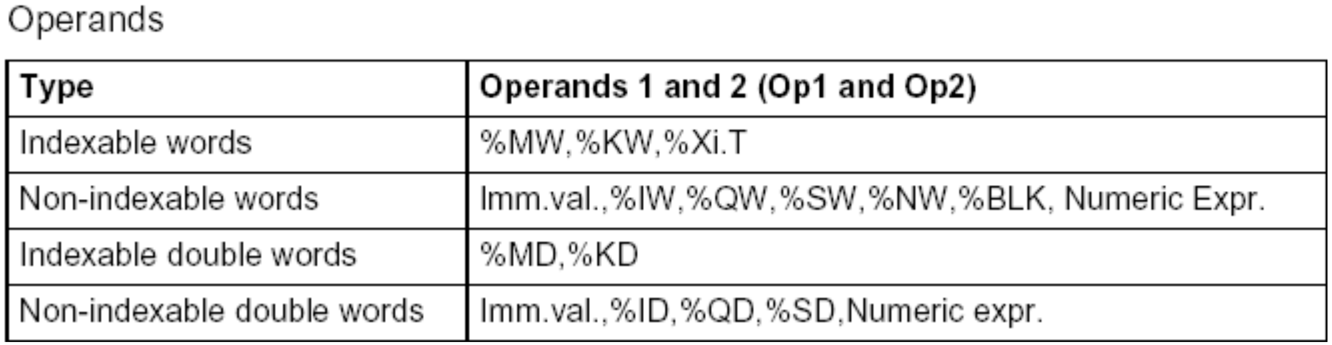


AND	AND (bit by bit) between two operands
OR	logical OR (bit by bit) between two operands
XOR	exclusive OR (bit by bit) between two operands
NOT	logical complement (bit by bit) of an operand



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

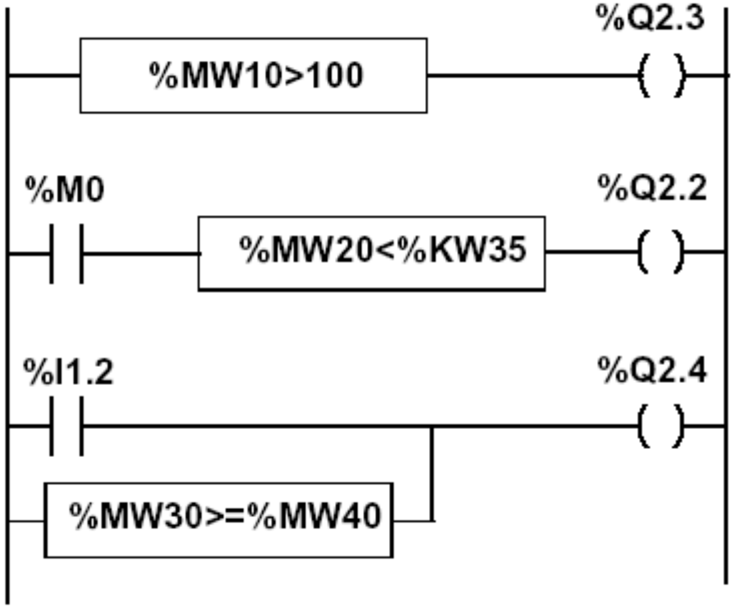
Type	**Operands 1 and 2 (Op1 and Op2)**
Indexable words	%MW,%KW,%Xi.T
Non-indexable words	Imm.val.,%IW,%QW,%SW,%NW,%BLK, Numeric Expr.
Indexable double words	%MD,%KD
Non-indexable double words	Imm.val.,%ID,%QD,%SD,Numeric expr.

Ladder diagram

Numerical Processing

Example:

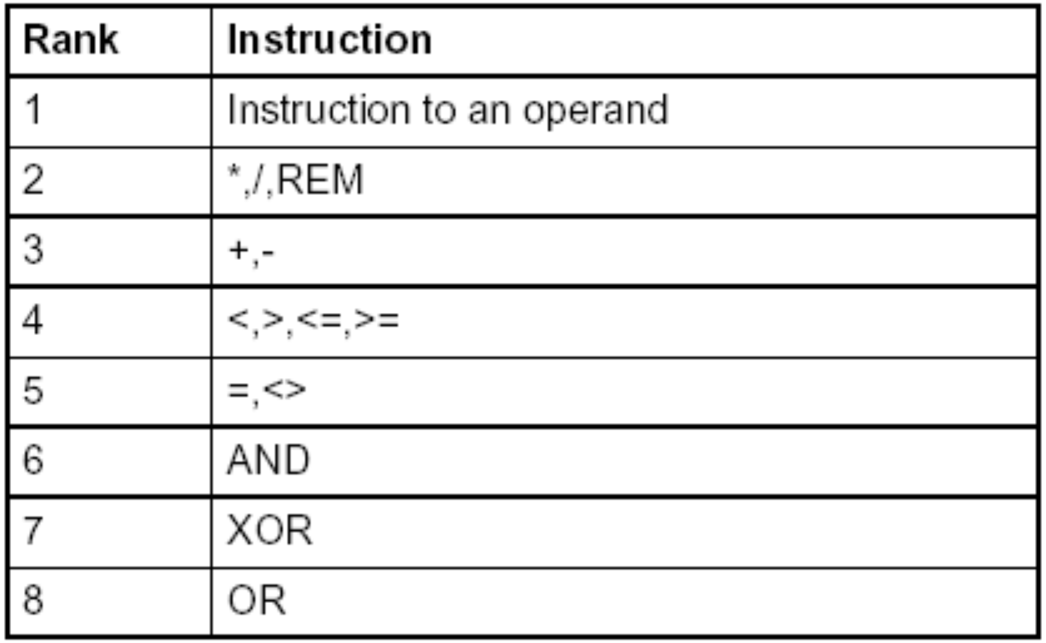
Logic functions



Ladder diagram

Numerical Processing

Priorities on the execution of the operations



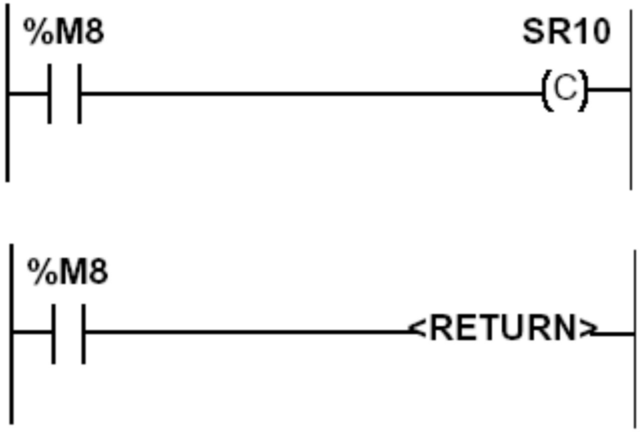
Rank	Instruction
1	Instruction to an operand
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

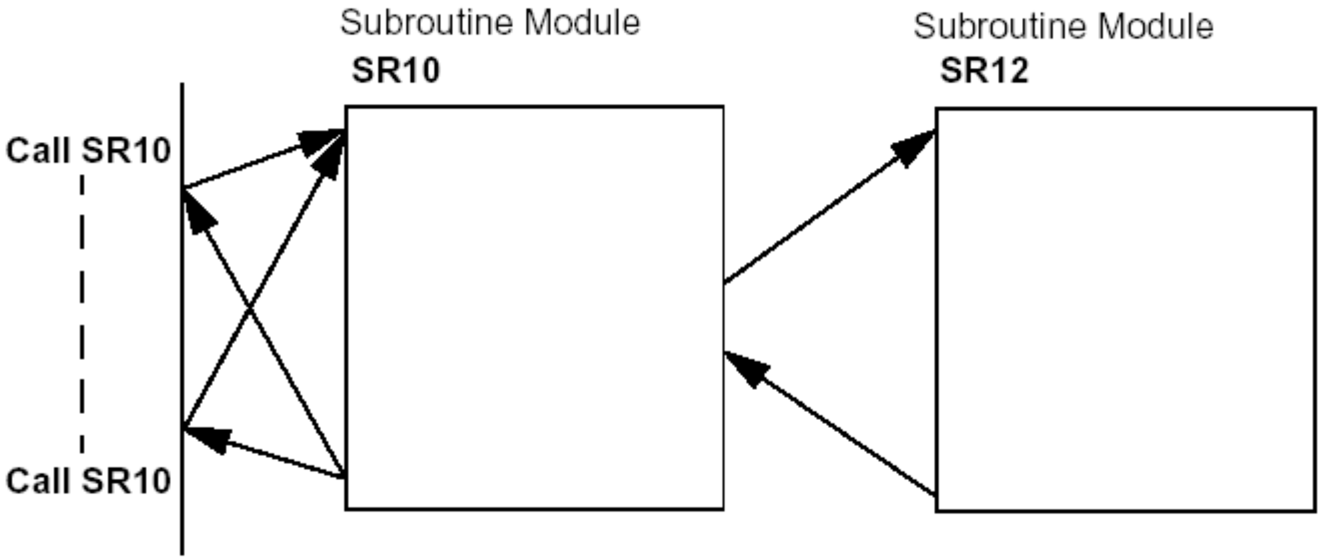
Ladder diagram

Structures for Control of Flux

Subroutines

Call and Return



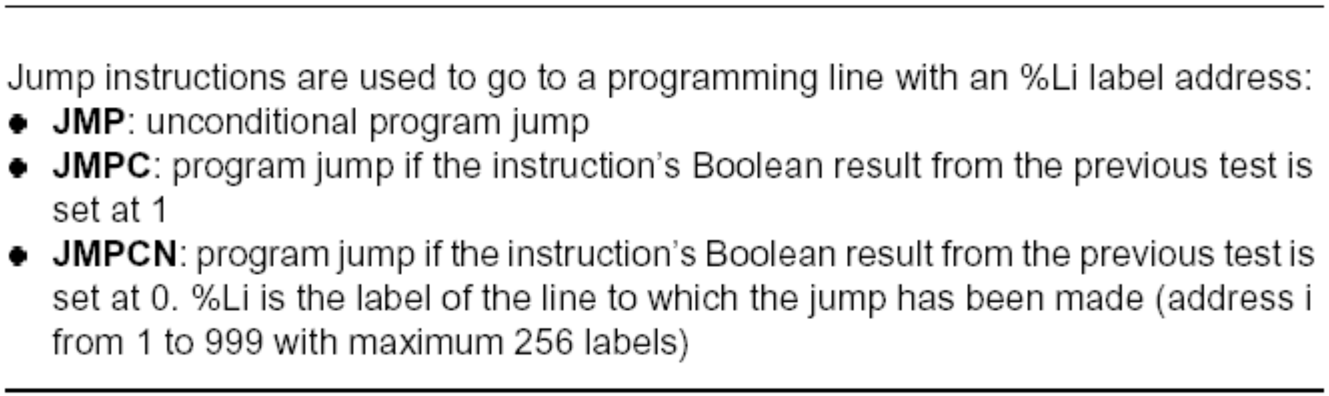


Ladder diagram

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

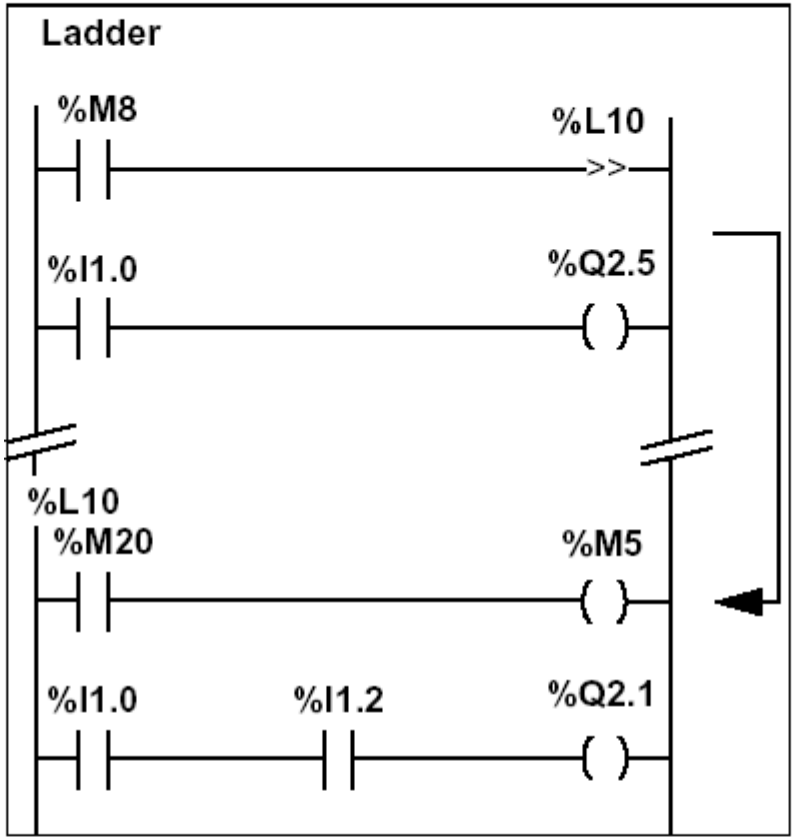
Structures for Control of Flux

Example:

Use of jump instructions

Attention to:

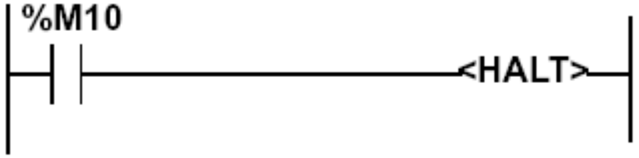
- **INFINITE LOOPS ...**
- **It is not a good style of programming!...**
- **Does not improove the legibility of the proposed solution.**



Ladder diagram

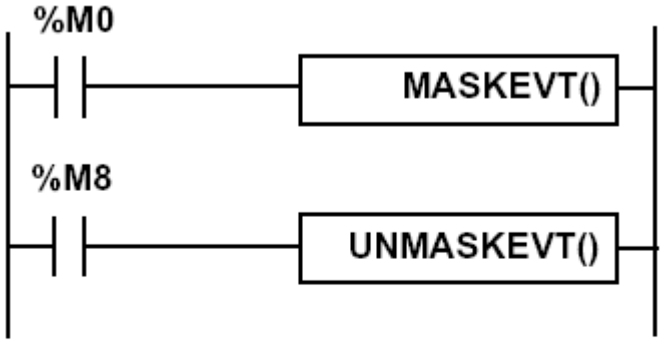
Structures for Control of Flux

Halt



Stops all processes!

Events masking



Ladder diagram

There are other advanced instrauctions (see manuel)

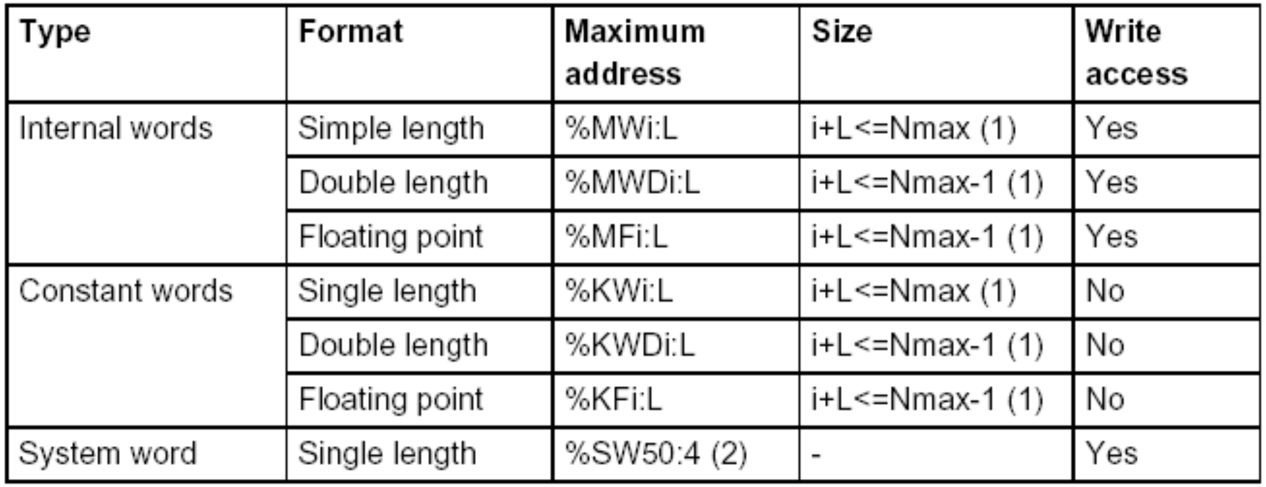
- **Monostable**
- **Registers of 256 words (LIFO ou FIFO)**
- ***DRUMs***
- **Comparators**
- ***Shift-registers***

...

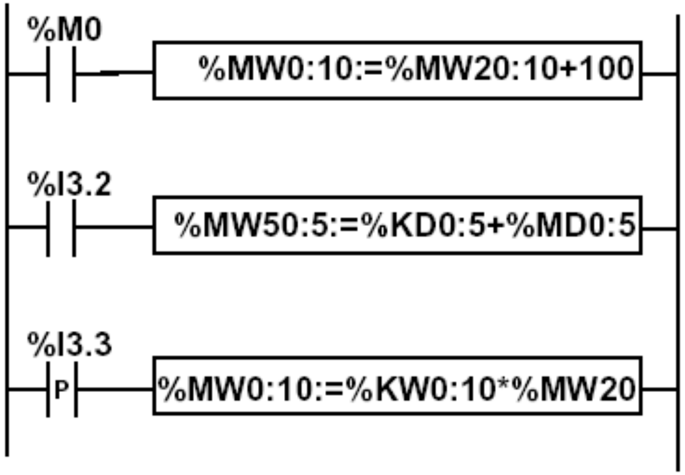
- **Functions to manipulate *floats***
- **Functions to convert bases and types**

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



Ladder diagram

Each PLC has limitations in terms of connections

Example:

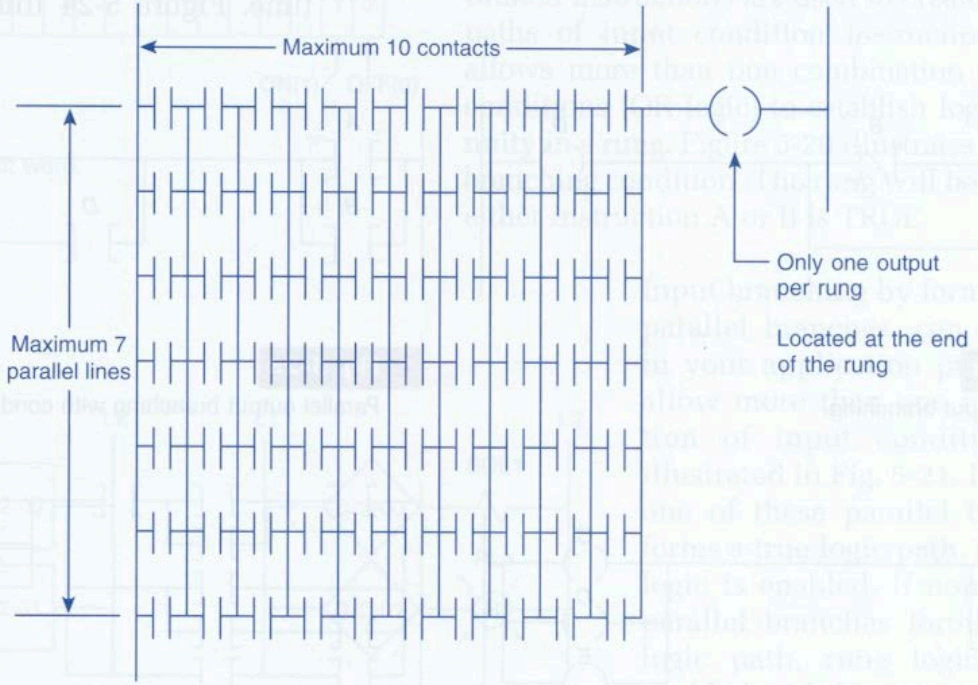
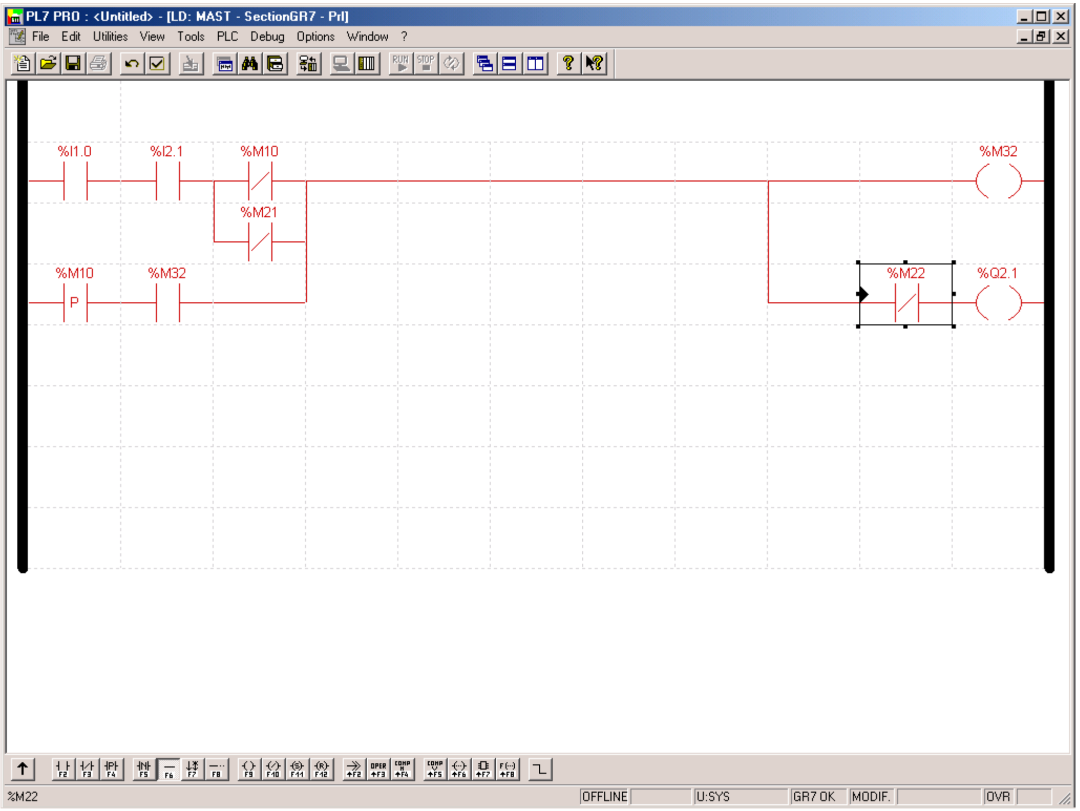


Fig. 5-27

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.

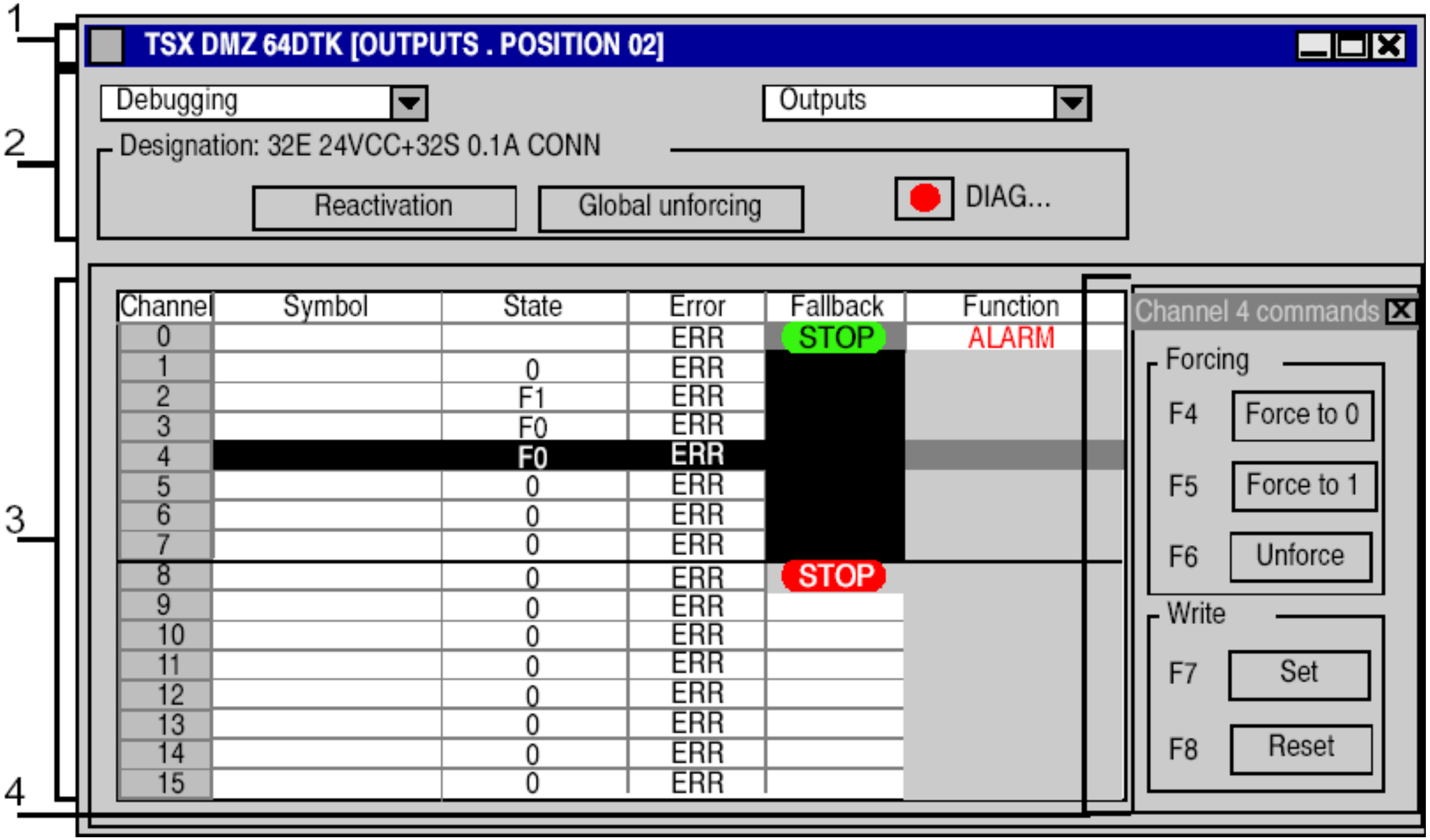
Ladder diagram

It is important to learn the potentialities and ...
the limitations of the developing tools,
i.e. "TO STUDY the manuals is a MUST."


PL7 PRO : <Untitled> - [LD: MAST - SectionGR7 - Prl]
File Edit Utilities View Tools PLC Debug Options Window ?
%I1.0
%I2.1
%M10
%M21
%M32
%M10
P
%M32
%M22
%Q2.1
%M22
OFFLINE
U:SYS
GR7 OK
MODIF.
OVR

Ladder diagram

Learn how to develop and debug programs (and how to do the fine tunning**)**.



TSX DMZ 64DTK [OUTPUTS . POSITION 02]

Debugging

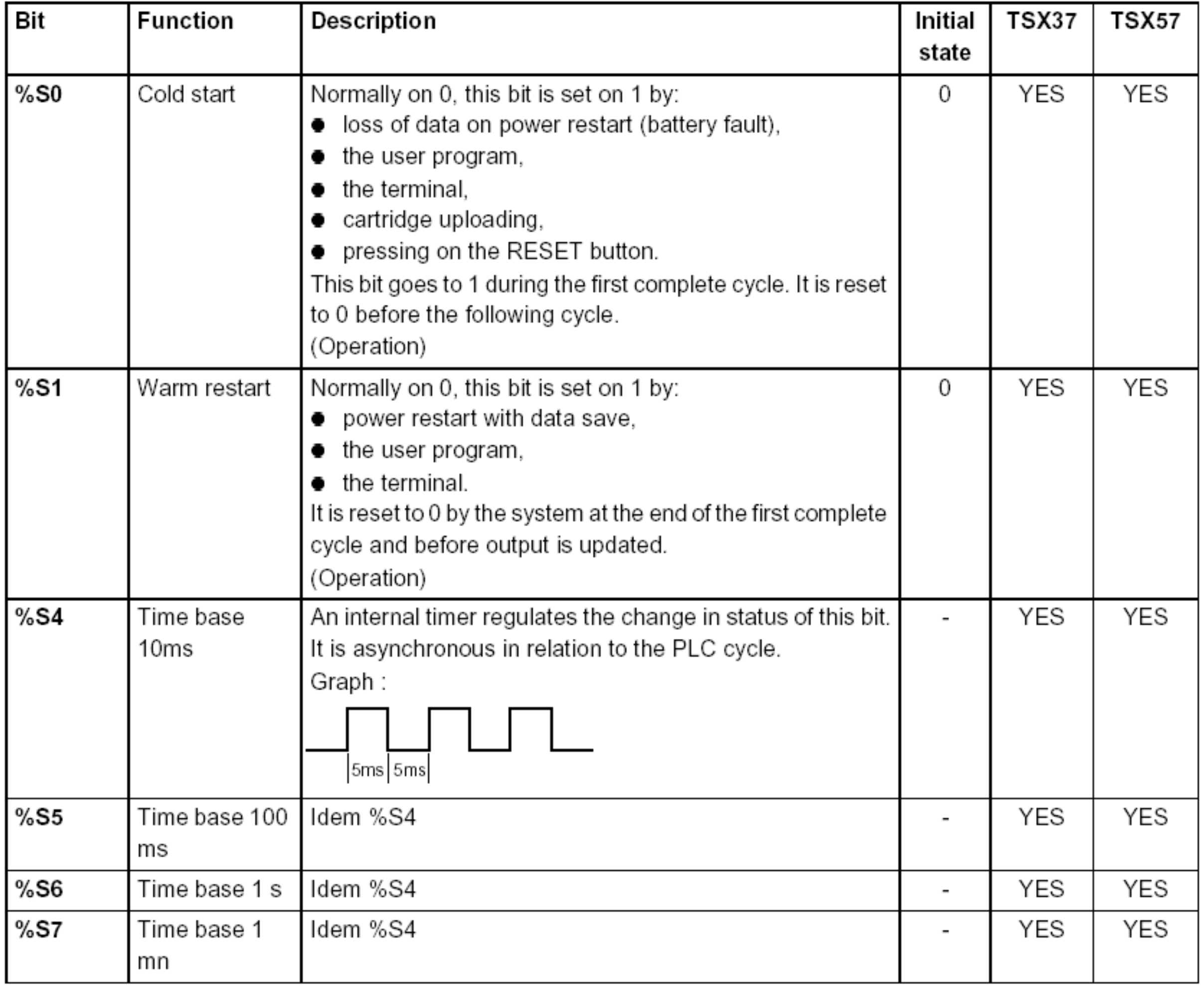
Outputs

Designation: 32E 24VCC+32S 0.1A CONN

Channel	Symbol	State	Error	Fallback	Function
0			ERR	STOP	ALARM
1		0	ERR		
2		F1	ERR		
3		F0	ERR		
4		F0	ERR		
5		0	ERR		
6		0	ERR		
7		0	ERR		
8		0	ERR	STOP	
9		0	ERR		
10		0	ERR		
11		0	ERR		
12		0	ERR		
13		0	ERR		
14		0	ERR		
15		0	ERR		

Ladder diagram

System information: system bits

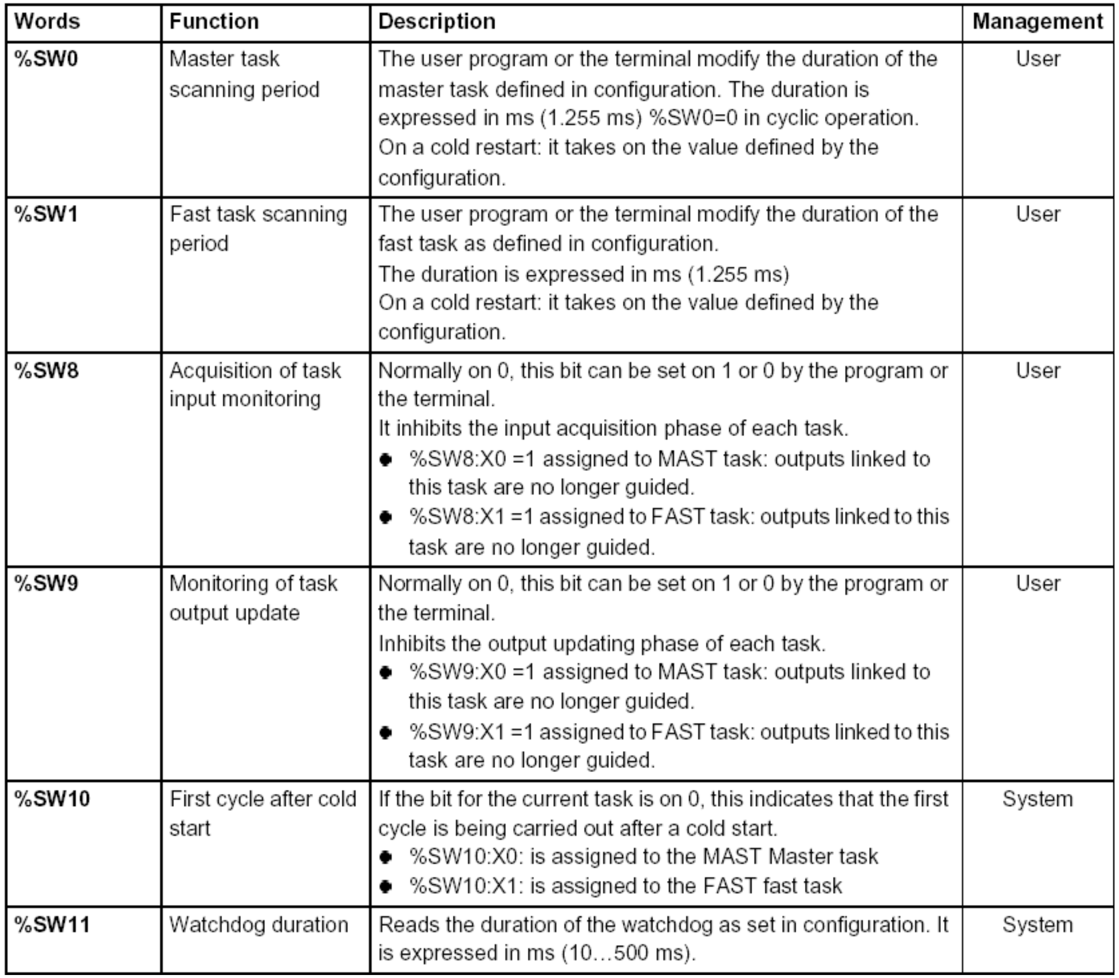


Bit	Function	Description	Initial state	TSX37	TSX57
%S0	Cold start	Normally on 0, this bit is set on 1 by: ● loss of data on power restart (battery fault), ● the user program, ● the terminal, ● cartridge uploading, ● pressing on the RESET button. This bit goes to 1 during the first complete cycle. It 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

Informação de Sistema: *words* de sistema



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

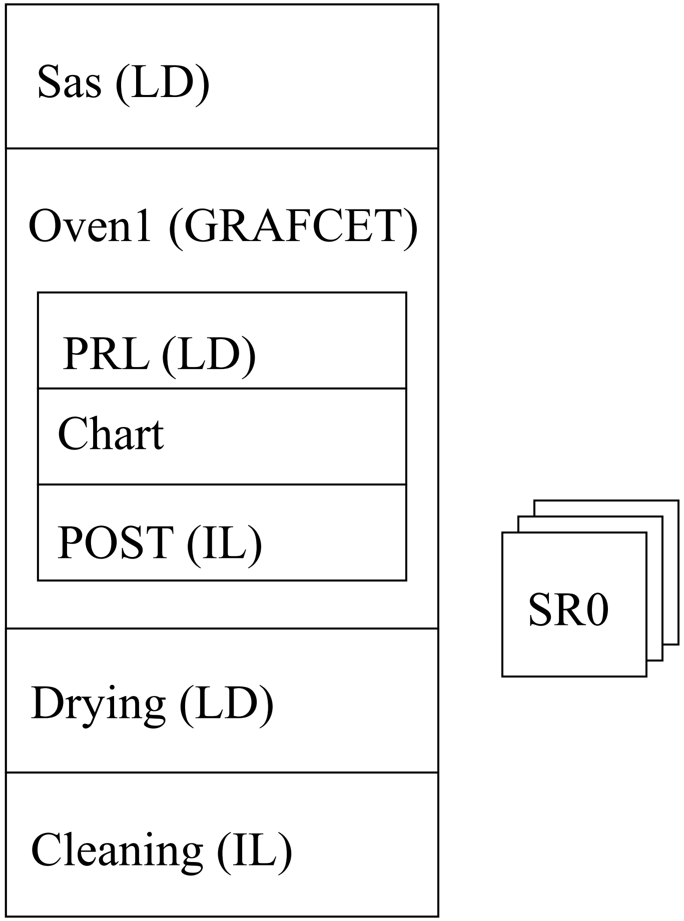
See manual for the remaining 140 words generated...

Ladder diagram

Software Organization

MAST – Master Task Program

Composed by sections



Execution

Cyclically

or

Periodically

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 a EV15)

Priority greater than MAST and FAST!

Event Generators

- **Inputs 0 to 3 in module 1, given transictions**
- **Counters**
- **Upon telegrams reception**
- **%S38 *Enables* or *disables* event processes**

(also with MASKEVT() or UNMASKEVT())

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*.

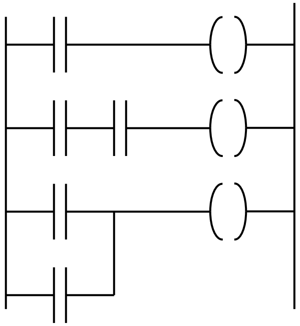
Software development resources.

...

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

PLCs Programming Languages (IEC 1131-3)

Ladder Diagram



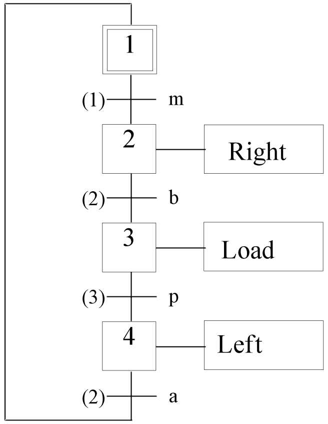
Structured Text

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

Instruction List

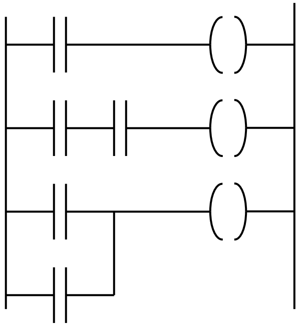
```
LD        %M12
AND       %I1.0
ANDN      %I1.1
OR        %M10
ST        %Q2.0
```

Sequential Function Chart (GRAFCET)



Linguagens de programação de PLCs (IEC 1131-3)

Ladder Diagram



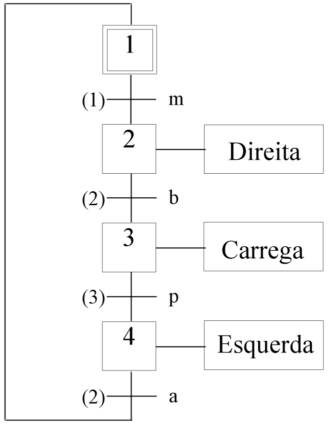
Structured Text

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

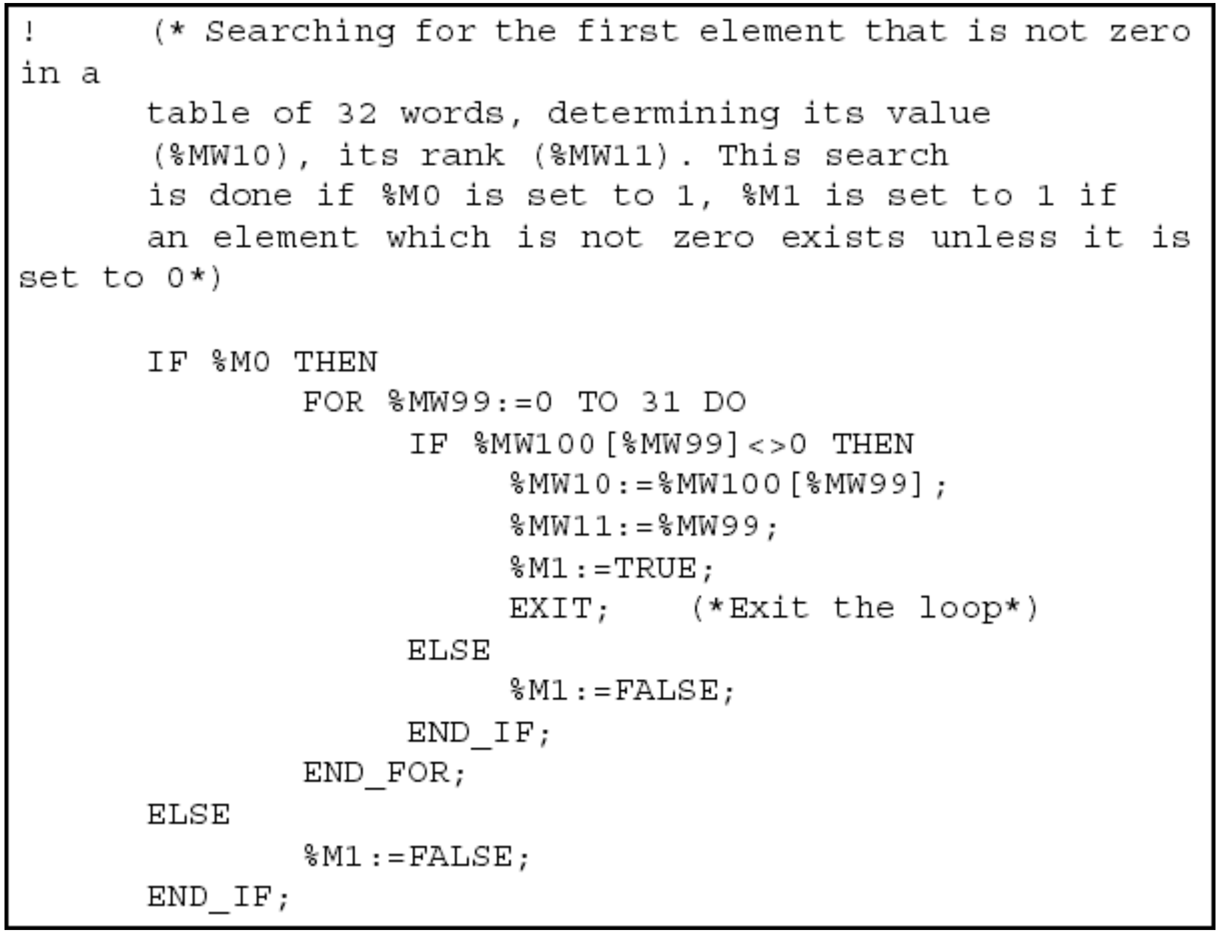
Instruction List

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

Sequential Function Chart (GRAFCET)



Structured Text

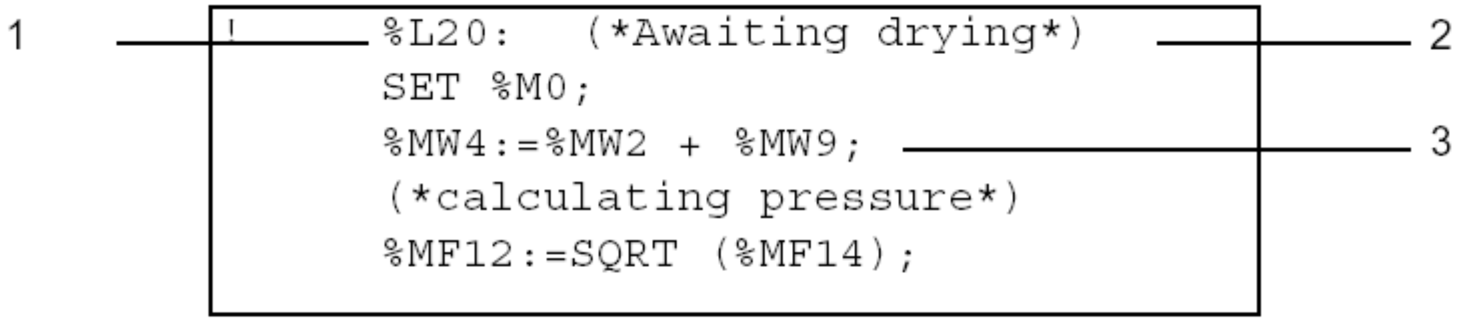


```
!     (* Searching for the first element that is not zero
in a
      table of 32 words, determining its value
      (%MW10), its rank (%MW11). This search
      is done if %M0 is set to 1, %M1 is set to 1 if
      an element which is not zero exists unless it is
set to 0*)

      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).



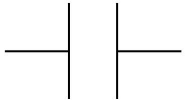
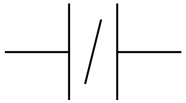
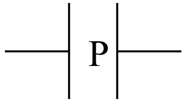
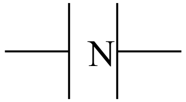
```
%L20:   (*Awaiting drying*)
SET %M0;
%MW4:=%MW2 + %MW9;
(*calculating pressure*)
%MF12:=SQRT (%MF14);
```

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.
:=NOT		Close contact: contacto is active (result is 1) while the control bit is 0.
:=RE		Contact in the rising edge: contact is active during a scan cycle where the control bit has a rising edge.
:=FE		Contact in the falling edge: contact is active during a scan cycle where the control bit has a falling edge.

Structured Text

Basic Instructions

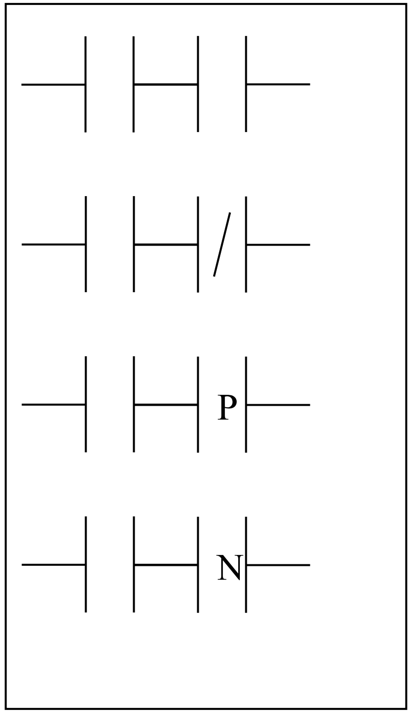
Store

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

Structured Text

Basic Instructions

AND

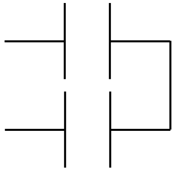
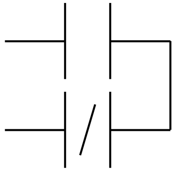
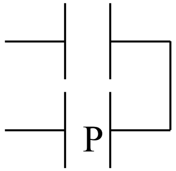
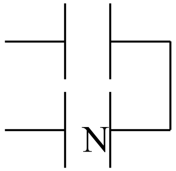


AND	AND of the operand with the result of the previous logical operation.
AND(NOT...)	AND of the operand with the inverted result of the previous logical operation.
AND(RE...)	AND of the rising edge with the result of the previous logical operation.
AND(FE...)	AND of the falling edge with the result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

Structured Text

Basic Instructions

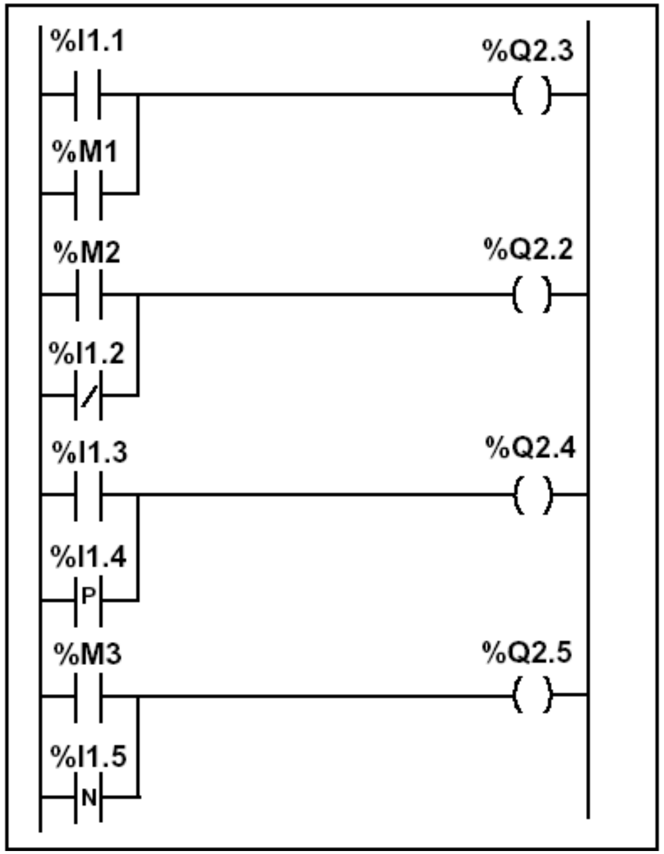
OR

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

Structured Text

Example:

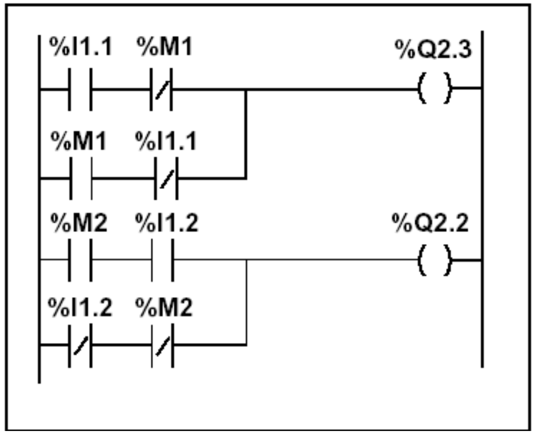
```
%Q2.3:=%I1.1 OR %M1;
%Q2.2:=%M2 OR (NOT%I1.2);
%Q2.4:=%I1.3 OR (RE%I1.4);
%Q2.5:=%M3 OR (FE%I1.5);
```

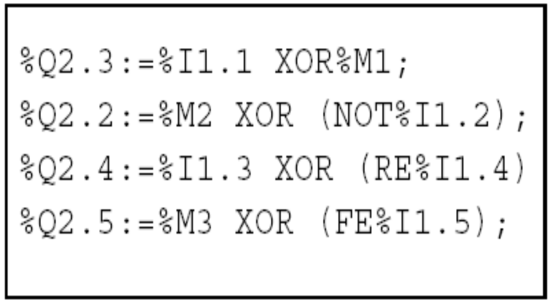


Structured Text

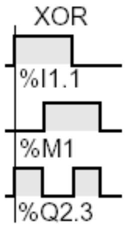
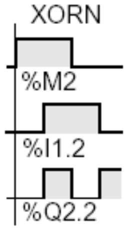
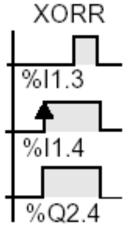
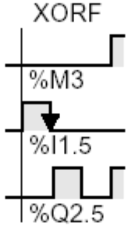
Basic Instructions

XOR



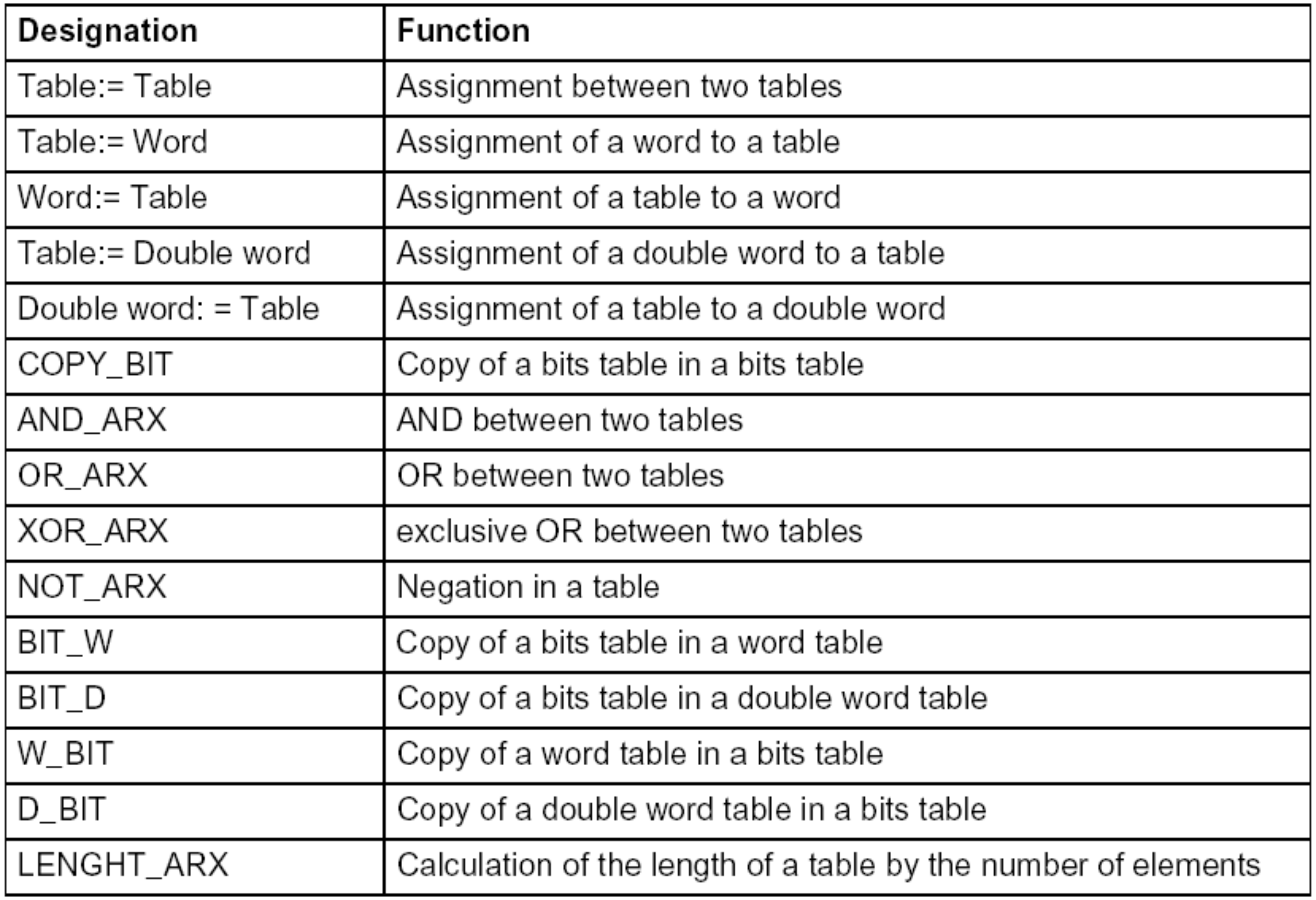


```
%Q2.3:=%I1.1 XOR%M1;
%Q2.2:=%M2 XOR (NOT%I1.2);
%Q2.4:=%I1.3 XOR (RE%I1.4)
%Q2.5:=%M3 XOR (FE%I1.5);
```

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

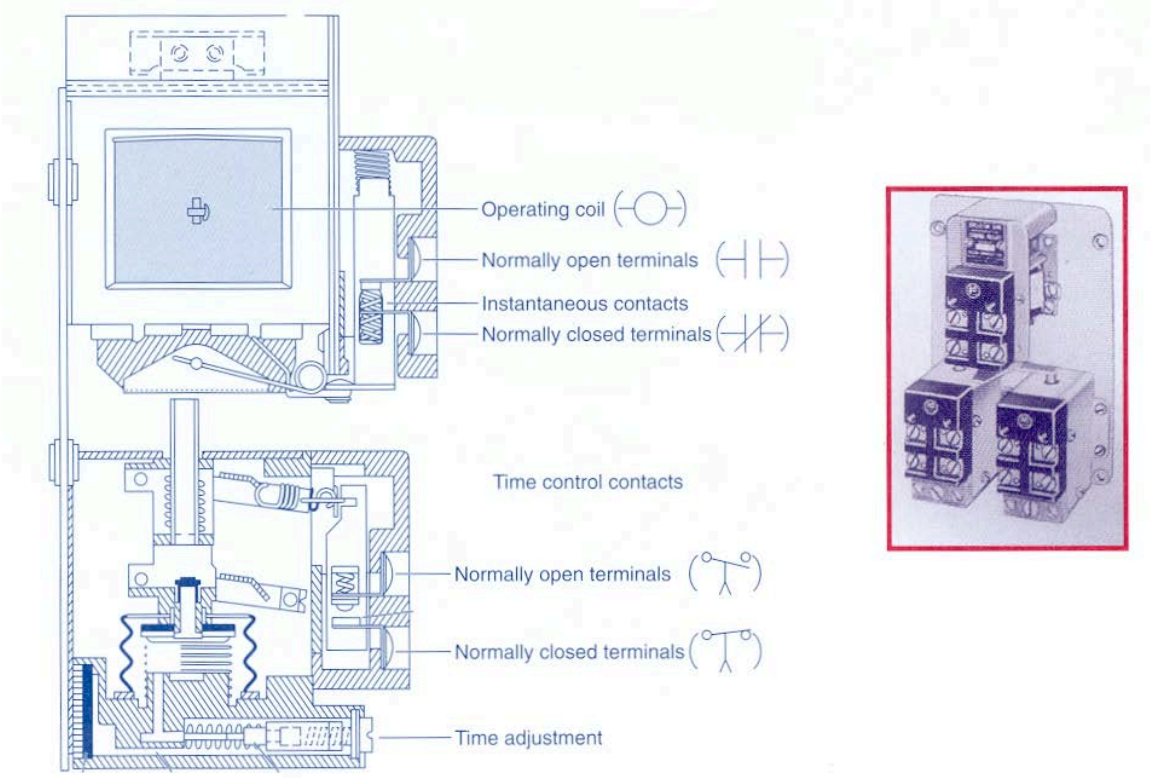


Fig. 7-1

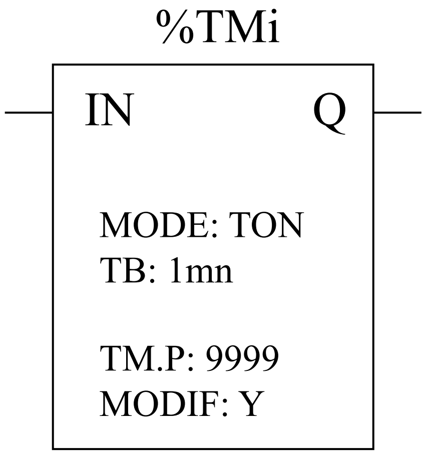
Pneumatic on-delay timer. *(Courtesy of Allen-Bradley Company, Inc.)*

Structured Text

Temporized Relays

or

Timers

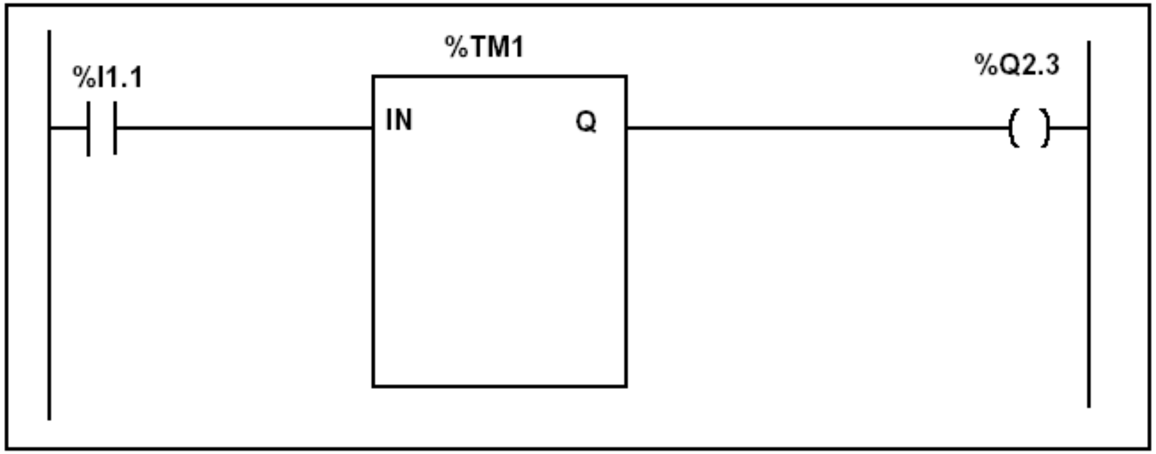


Characteristics:

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

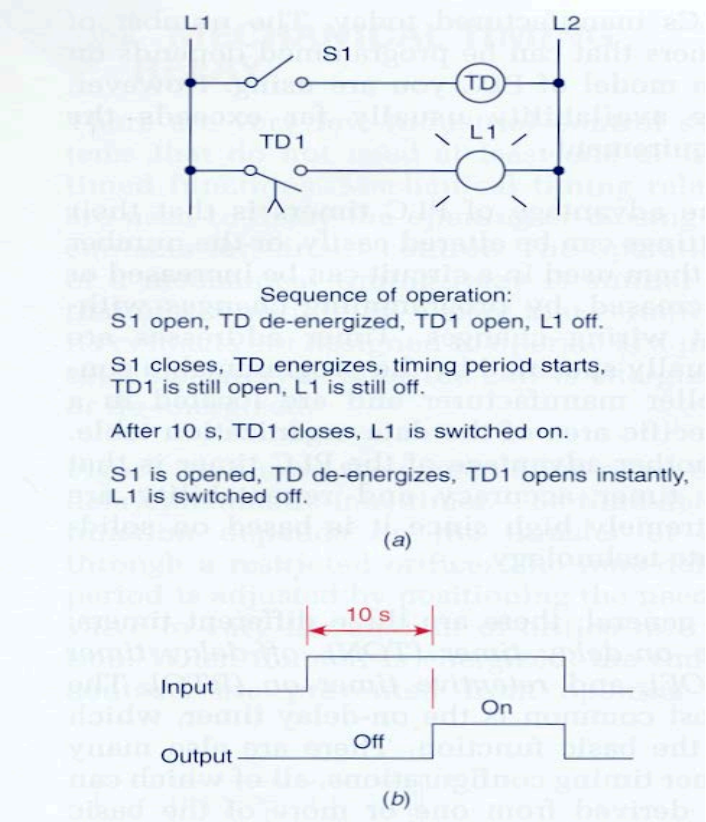
Relés temporizados
Ou
Timers



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

Structured Text

Example:



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.

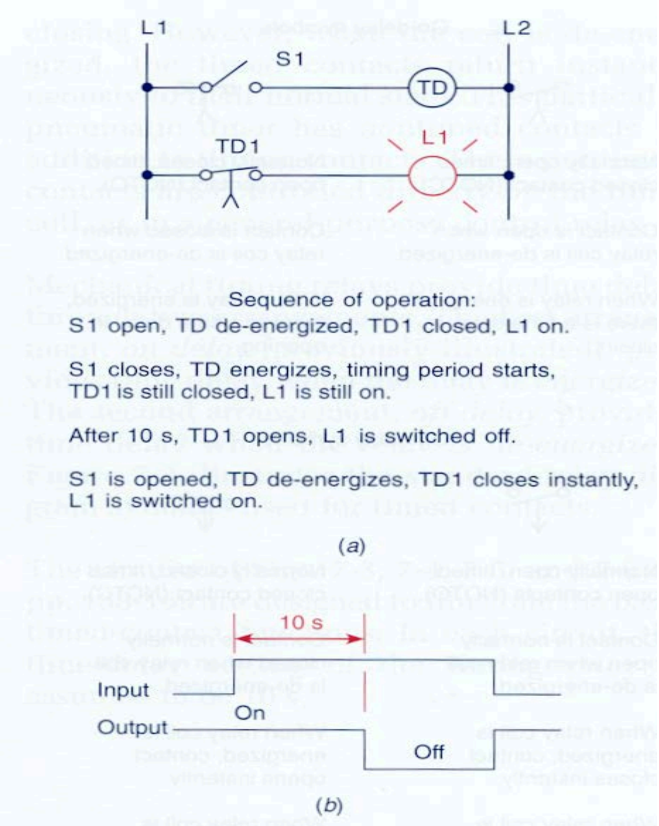
S1 is opened, TD de-energizes, TD1 opens instantly, L1 is switched off.

(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)

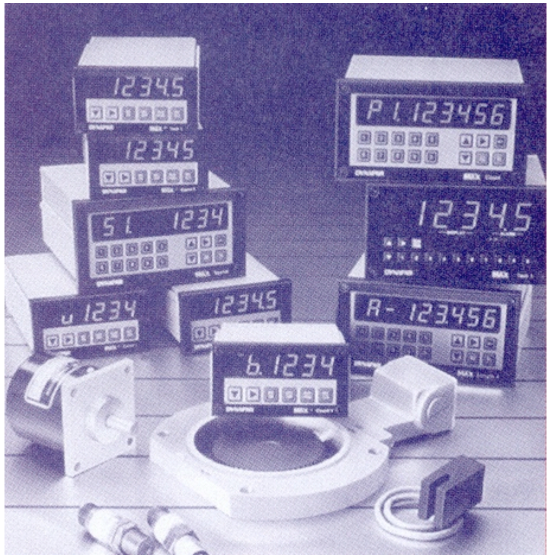
(b)

Fig. 7-4

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

Structured Text

Counters



Some applications...

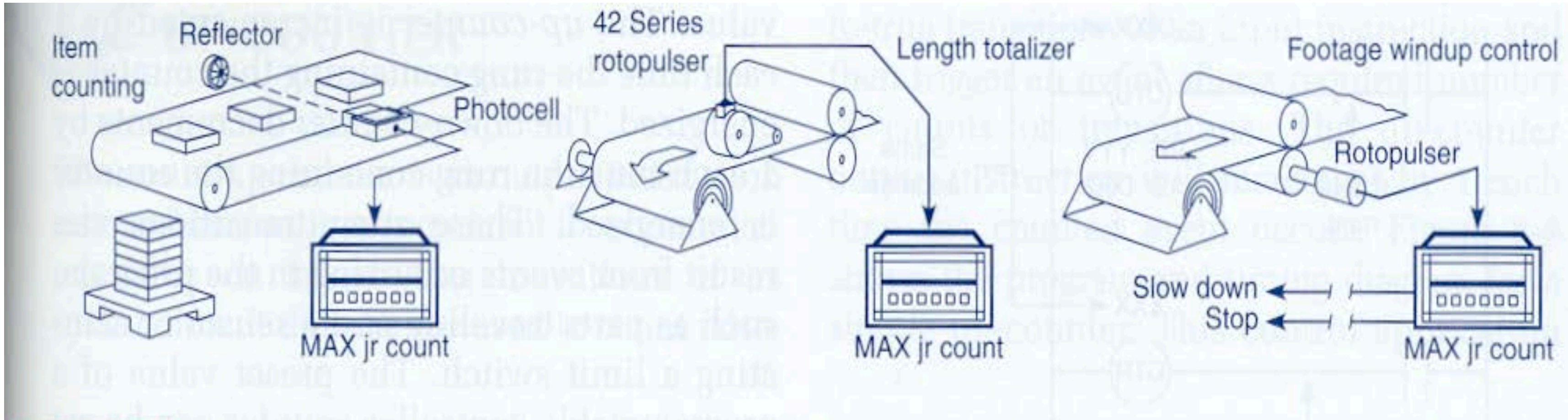
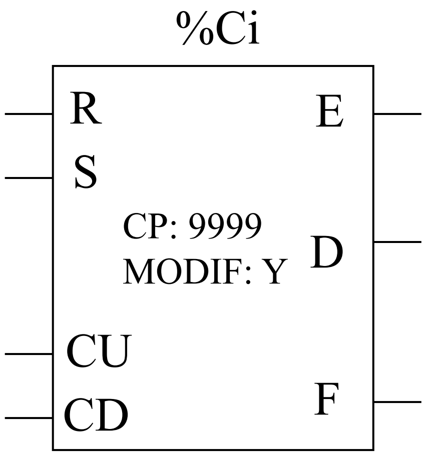


Fig. 8-3

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

Structured Text

Counters



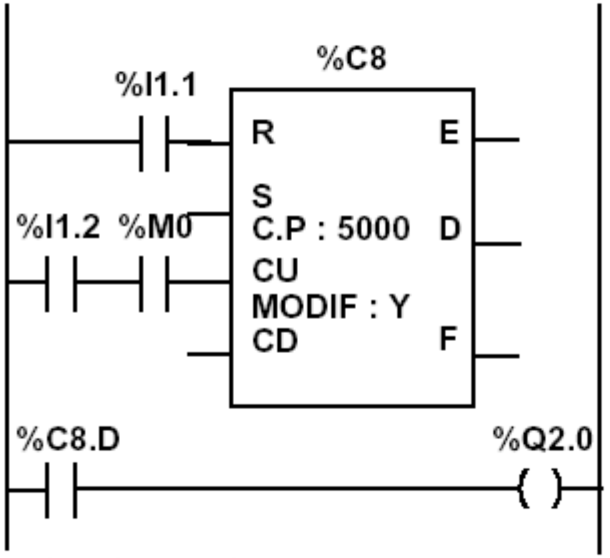
Characteristics:

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;
```

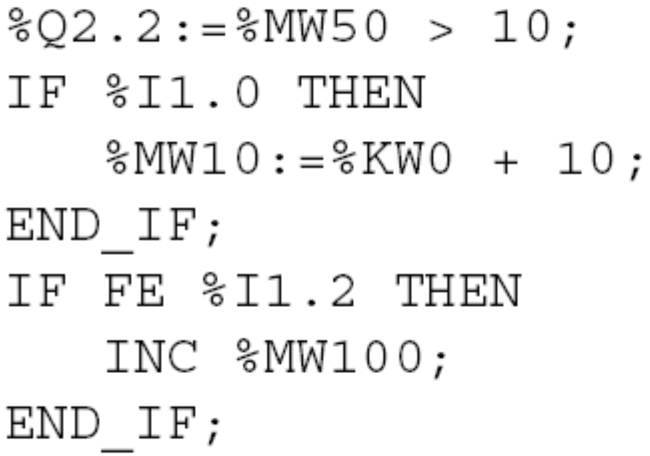
Instruction list language

```
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

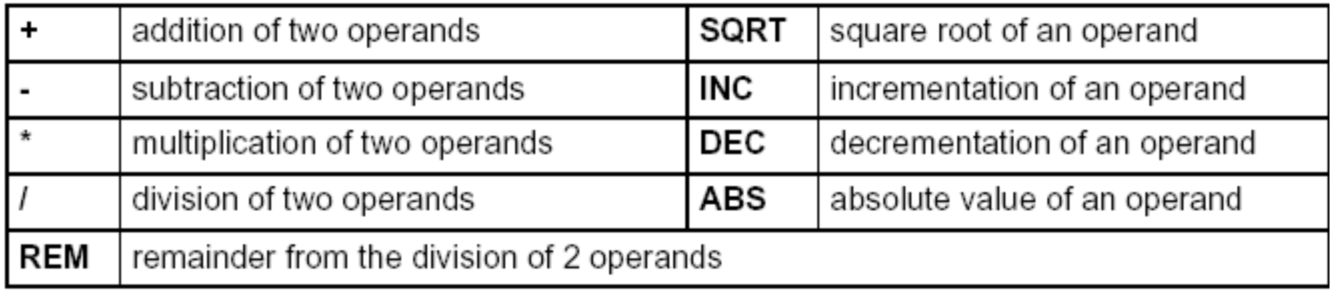


```
%Q2.2:=%MW50 > 10;
IF %I1.0 THEN
   %MW10:=%KW0 + 10;
END_IF;
IF FE %I1.2 THEN
   INC %MW100;
END_IF;
```

Structured Text

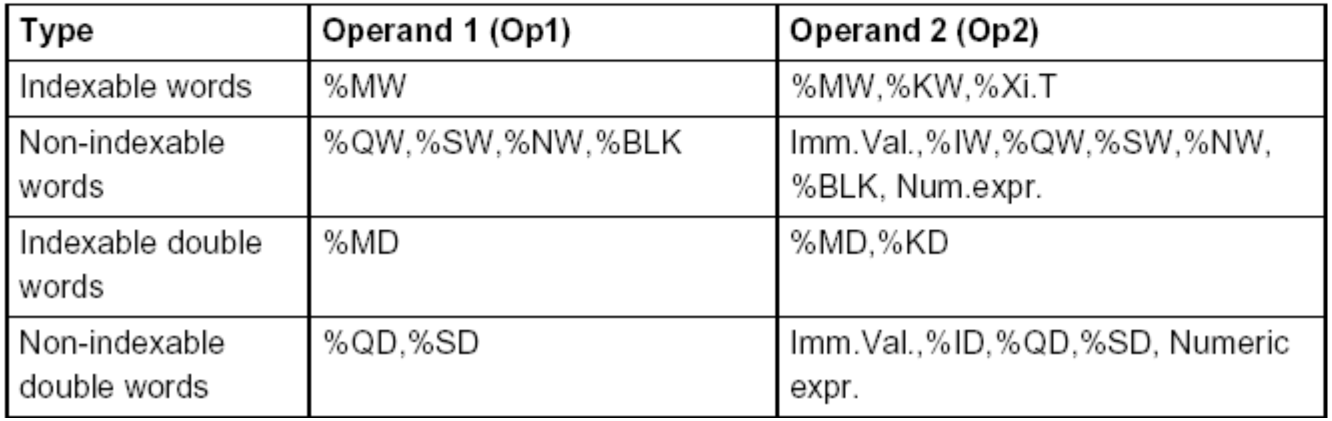
Numerical Processing

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.

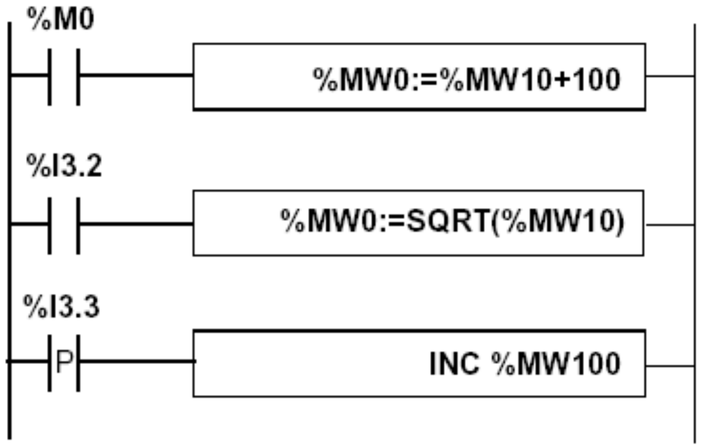
Structured Text

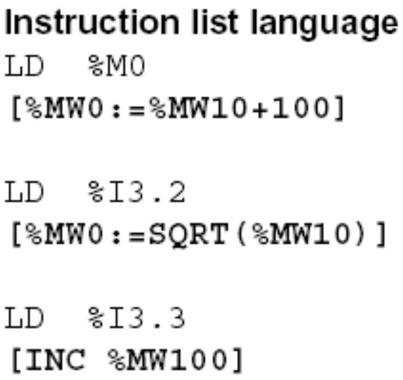
Numerical Processing

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

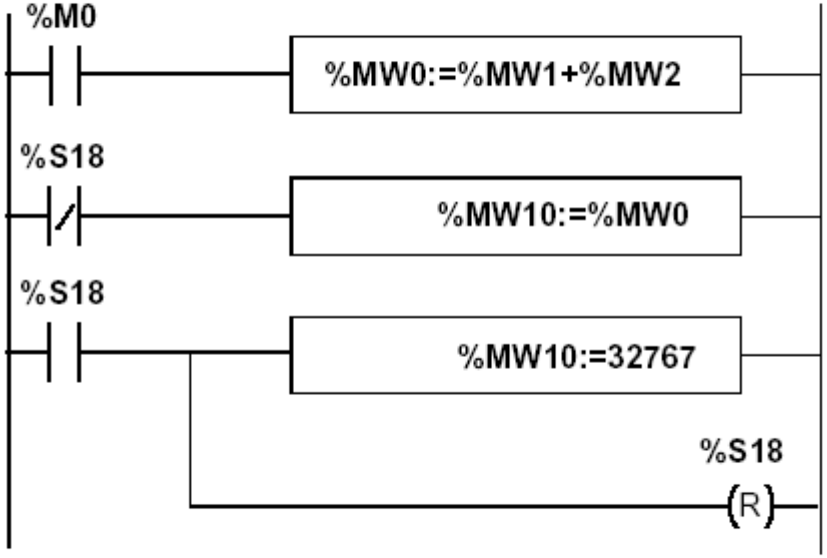
```
LD  %M0
[%MW0:=%MW10+100]

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;
```

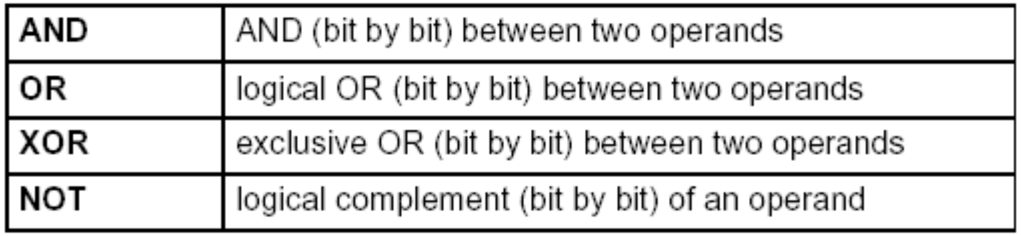
Example in instruction list language:

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

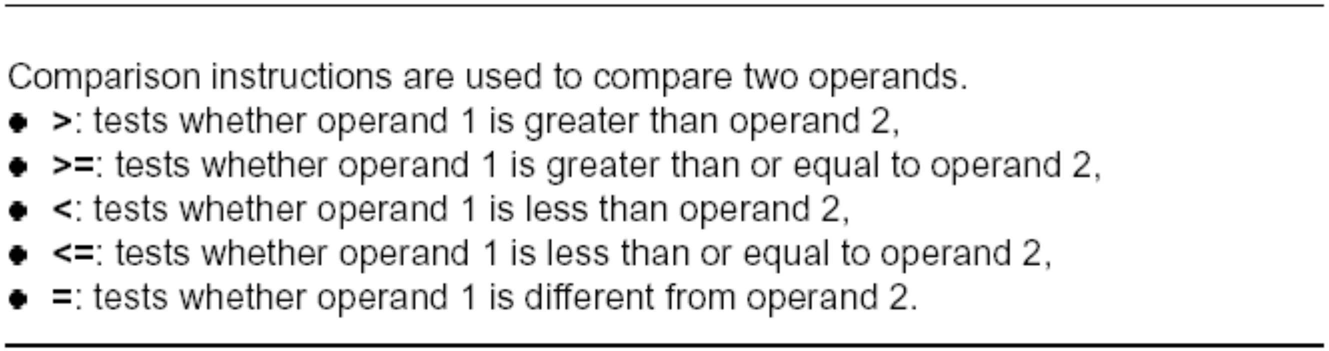
Structured Text

Numerical Processing

Logic Functions

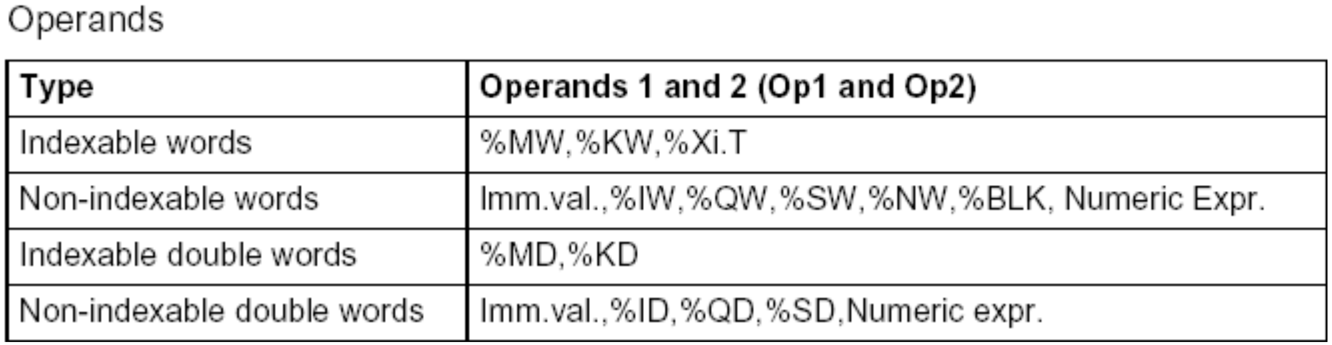


AND	AND (bit by bit) between two operands
OR	logical OR (bit by bit) between two operands
XOR	exclusive OR (bit by bit) between two operands
NOT	logical complement (bit by bit) of an operand



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

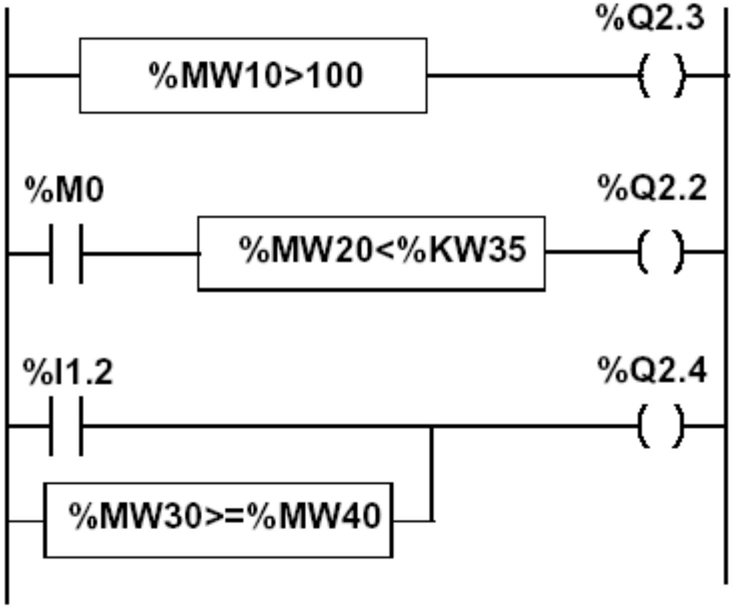
Type	Operands 1 and 2 (Op1 and Op2)
Indexable words	%MW,%KW,%Xi.T
Non-indexable words	Imm.val.,%IW,%QW,%SW,%NW,%BLK, Numeric Expr.
Indexable double words	%MD,%KD
Non-indexable double words	Imm.val.,%ID,%QD,%SD,Numeric expr.

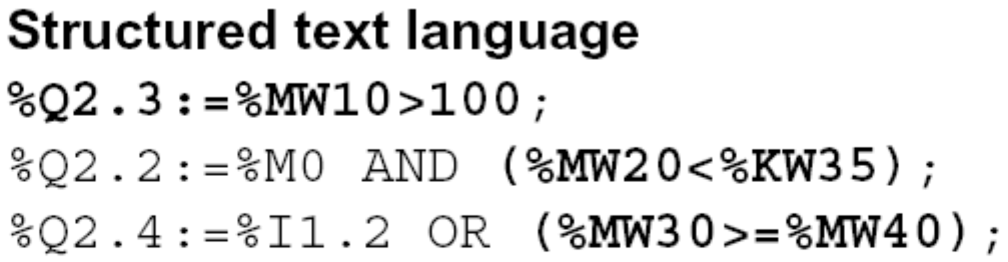
Structured Text

Numerical Processing

Example:

Logic functions





Structured text language

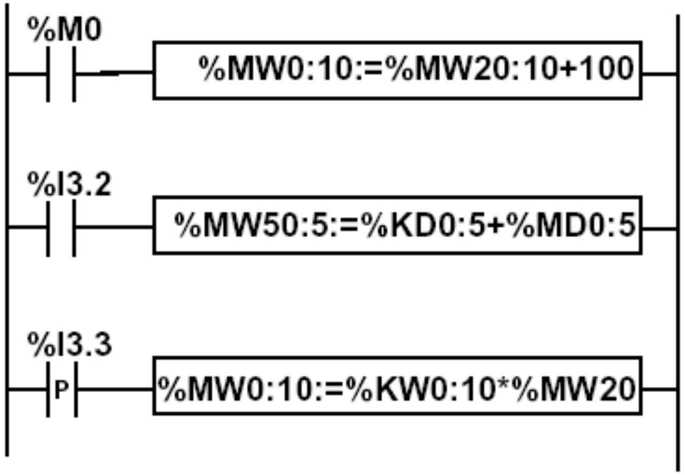
```
%Q2.3:=%MW10>100;
%Q2.2:=%M0 AND (%MW20<%KW35);
%Q2.4:=%I1.2 OR (%MW30>=%MW40);
```

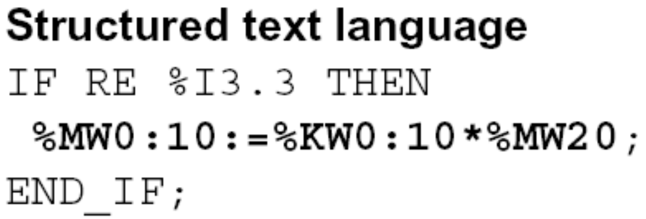
Structured Text

Numerical Processing

Example:

Numeric Tables Manipulation





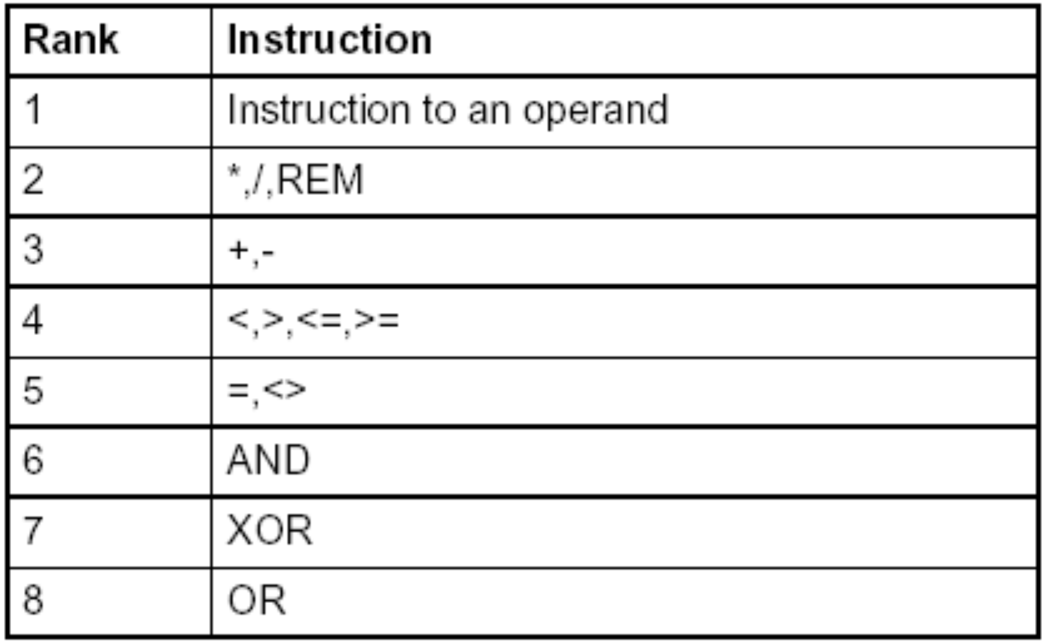
Structured text language

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

Structured Text

Numerical Processing

Priorities on the execution of the operations



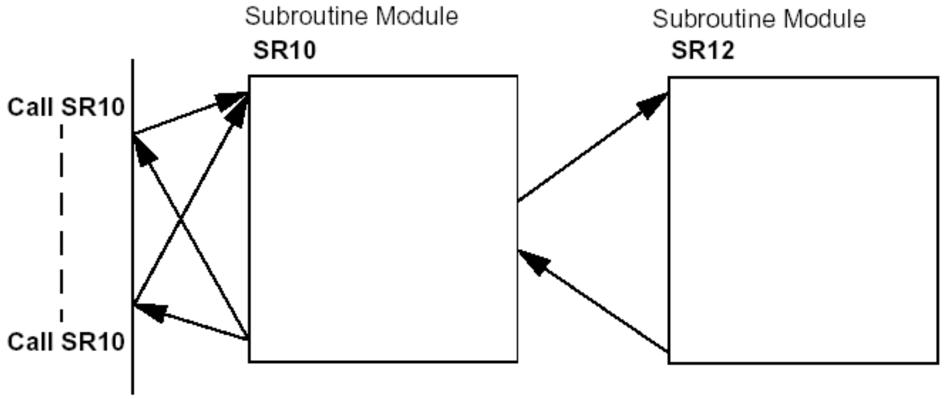
Rank	Instruction
1	Instruction to an operand
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

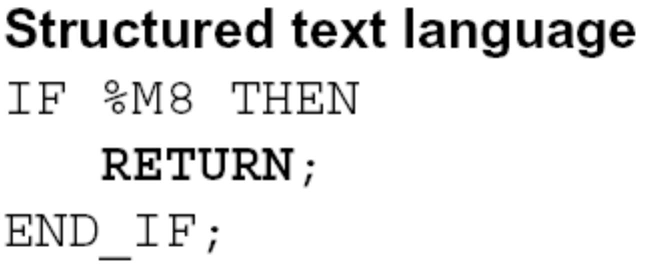
Structured Text

Structures for Control of Flux

Subroutines

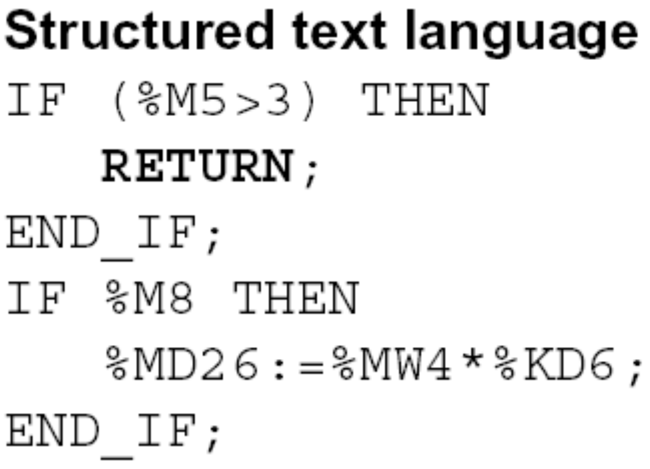
Call and Return





Structured text language

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



Structured text language

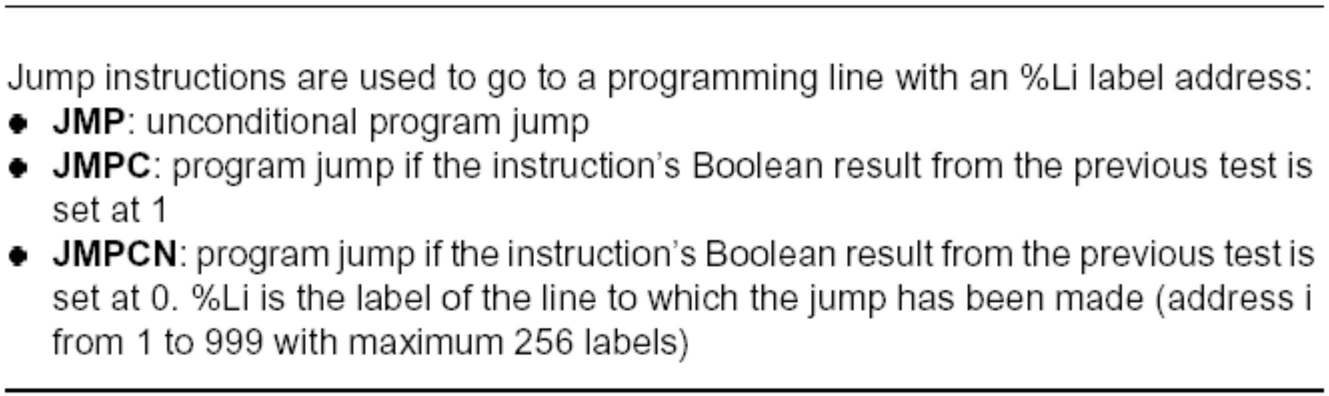
```
IF (%M5>3) THEN
   RETURN;
END_IF;
IF %M8 THEN
   %MD26:=%MW4*%KD6;
END_IF;
```

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:

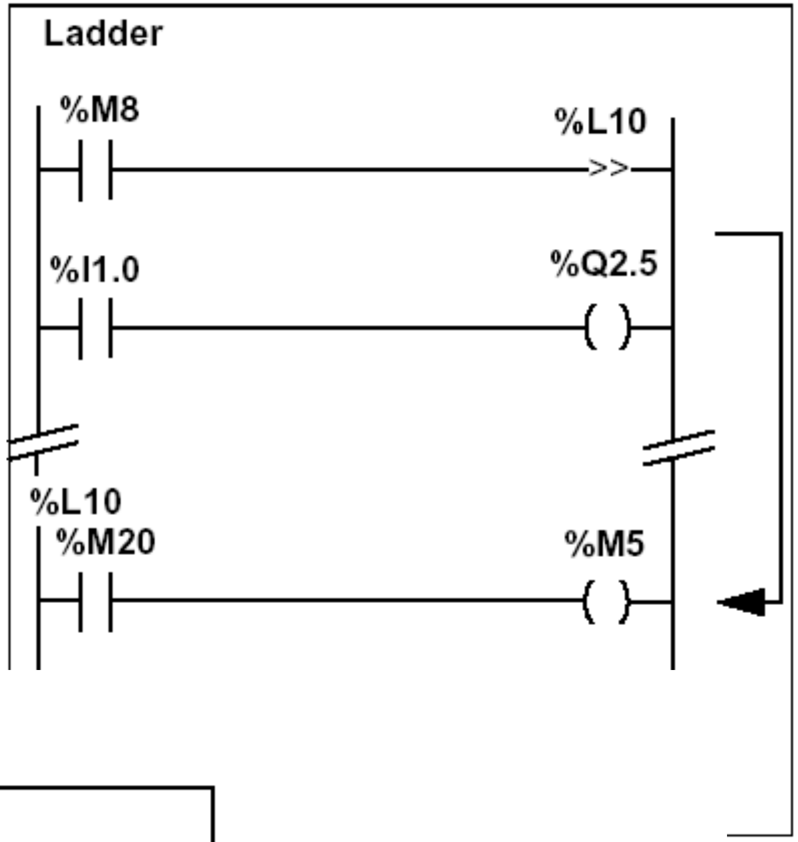
- **JMP**: unconditional program jump
- **JMPC**: program jump if the instruction's Boolean result from the previous test is set at 1
- **JMPCN**: program jump if the instruction's Boolean result from the previous test is set at 0. %Li is the label of the line to which the jump has been made (address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

Structured Text

Structures for Control of Flux

Example:

Use of jump instructions



Attention to:

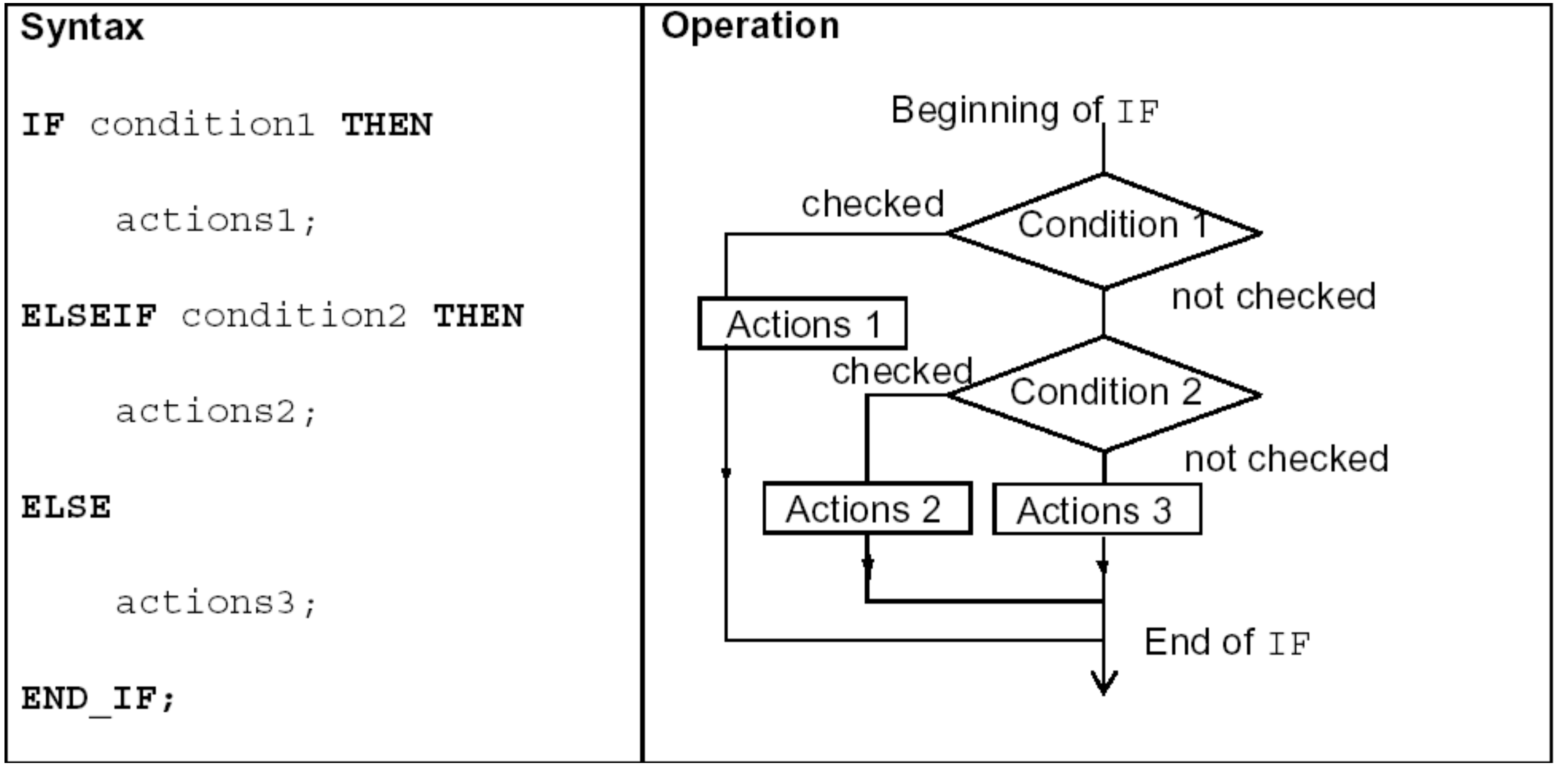
```
IF %M8 THEN
   JUMP %L10;
END_IF;
%Q2.5:=%I1.0;

------
%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

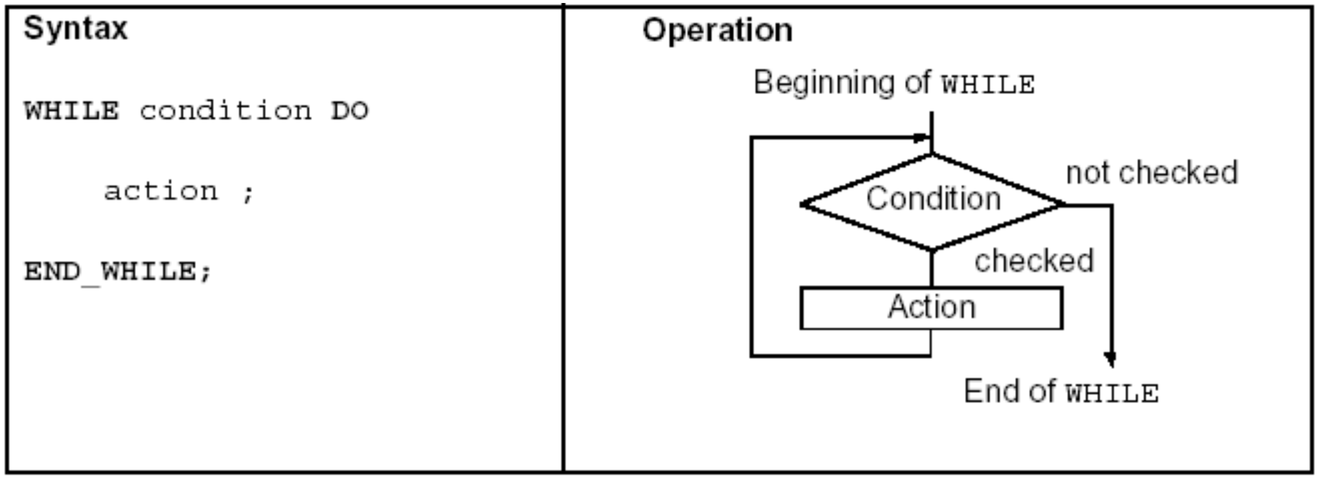
```
IF condition1 THEN
    actions1;
ELSEIF condition2 THEN
    actions2;
ELSE
    actions3;
END_IF;
```

Operation

Structured Text

Structures for Control of Flux

WHILE

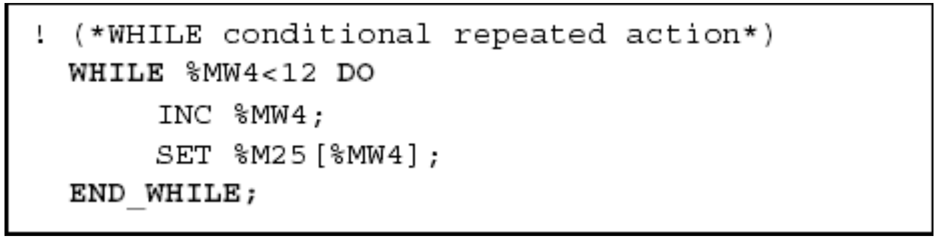


Syntax

```
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

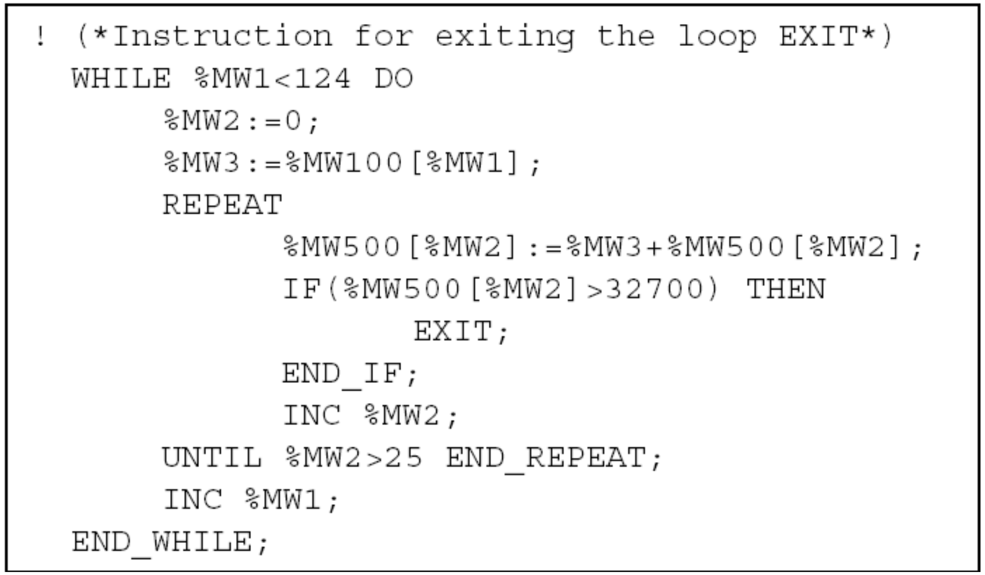
Structures for Control of Flux

REPEAT ... UNTIL

FOR ... DO

EXIT to abort the execution of a st6ructured flux control instruction

Example:

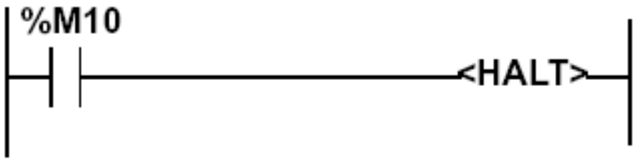


```
! (*Instruction for exiting the loop EXIT*)
  WHILE %MW1<124 DO
       %MW2:=0;
       %MW3:=%MW100[%MW1];
       REPEAT
              %MW500[%MW2]:=%MW3+%MW500[%MW2];
              IF(%MW500[%MW2]>32700) THEN
                     EXIT;
              END_IF;
              INC %MW2;
       UNTIL %MW2>25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

Structured Text

Structures for Control of Flux

Halt

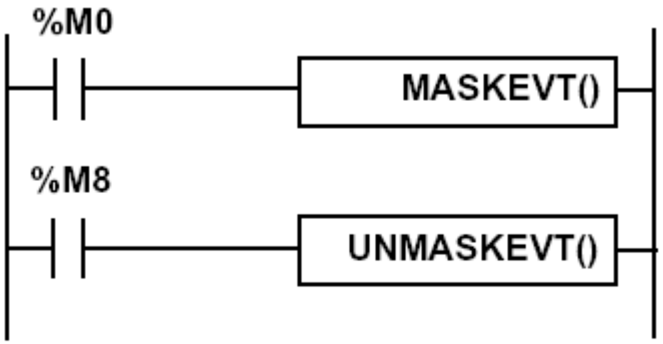


Structured text language

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

Stops all processes!

Events masking

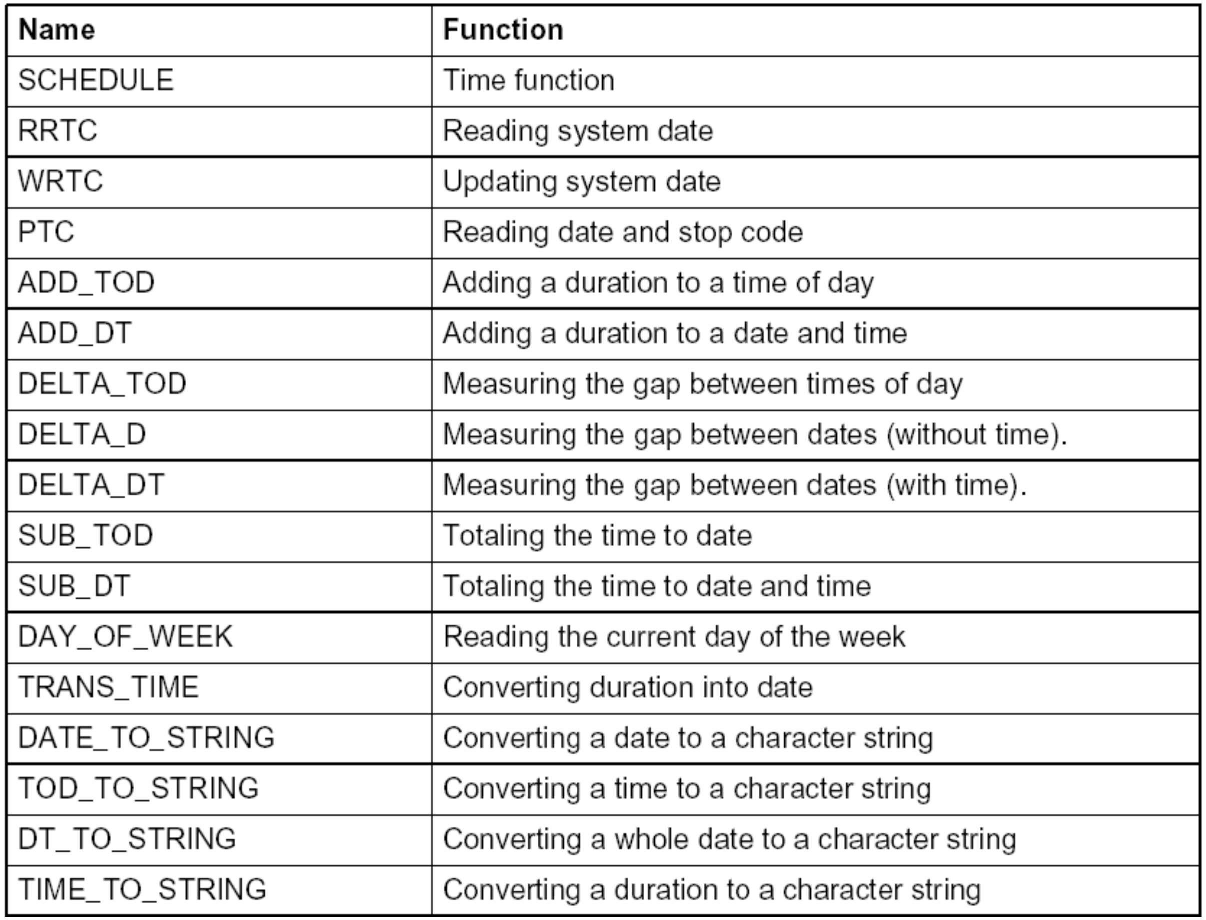


Structured text language

```
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***

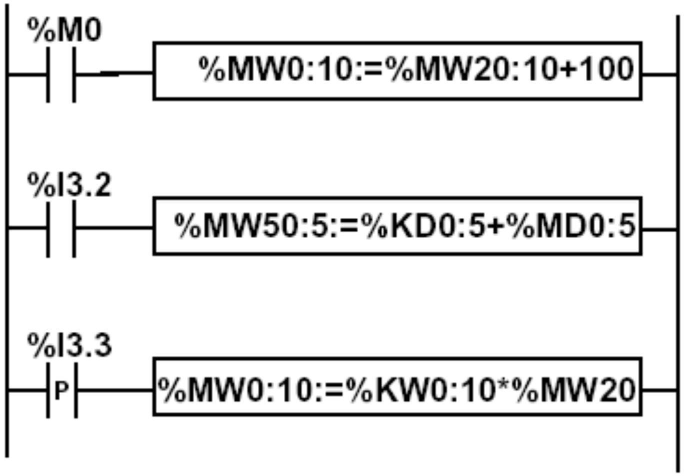
...

- **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]
```

Industrial Automation

(Automação de Processos Industriais)

GRAFCET

(Sequential Function Chart)

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. 3 – PLCs Programming Languages [2 weeks]

...

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

The GRAFCET norm.

Elements of the language.

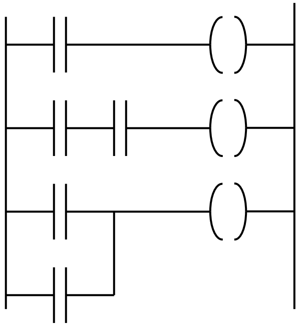
Modelling techniques using GRAFCET.

...

Chap. 5 – CAD/CAM and CNC Machines [1 week]

PLCs Programming Languages (IEC 1131-3)

Ladder Diagram



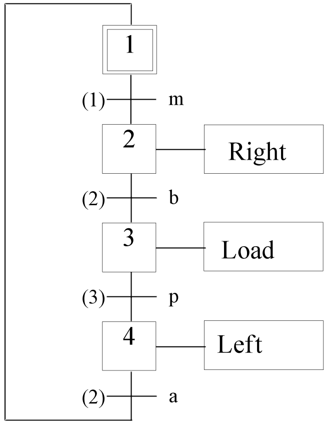
Structured Text

```
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)



Some pointers to GRAFCETs (SFCs)

History:	http://www.ecsi.org/ecsi/Doc/OtherDoc/SLDL/PDF/caspi.pdf http://www.lurpa.ens-cachan.fr/grafcet/groupe/gen_g7_uk/geng7.html
Tutorial:	http://asi.insa-rouen.fr/~amadisa/grafcet_homepage/tutorial/index.html http://www-ipst.u-strasbg.fr/pat/autom/grafce_t.htm
Simulator:	http://asi.insa-rouen.fr/~amadisa/grafcet_homepage/grafcet.html http://www.automationstudio.com (See projects)
Bibliography:	* Petri Nets and GRAFCET: Tools for Modelling Discrete Event Systems R. DAVID, H. ALLA, New York : PRENTICE HALL Editions, 1992 * Programação de Autómatos, Método GRAFCET, José Novais, Fundação Calouste Gulbenkian * Norme Française NF C 03-190 + R1 : Diagramme fonctionnel "GRAFCET" pour la description des systèmes logiques de commande
Homepage:	http://www.lurpa.ens-cachan.fr/grafcet/

History

GRAFCET

- 1975 – Decision of the workgroup "Logical Systems" da AFCET (Association Française de Cybernétique Economique et Technique) on the creation of a committee to study a standard for the representation of logical systems and automation.

- 1977 – GRAFCET definition (Graphe Fonctionnel de Commande Etape-Transition).

- 1979 – Dissemination in schools and adopted as research area for the implementation of solutions of automation in the industry.

- 1988 - GRAFCET becomes an international standard denominated as "Sequential Function Chart“, pela I.E.C.

GRAFCET

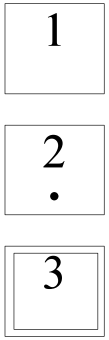
Basic Elements

Steps

Inactive

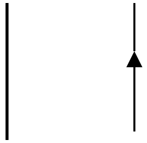
Active

Initial



Connections

Directed
Arc



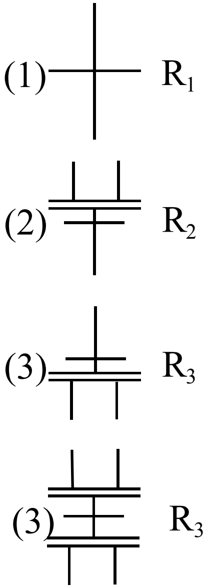
Transitions

Simple

Joint

Fork

Joint e *fork*



A**ctions** can be associated with **Steps**.

A logical receptivity function can be associated with each **Transition**.

GRAFCET

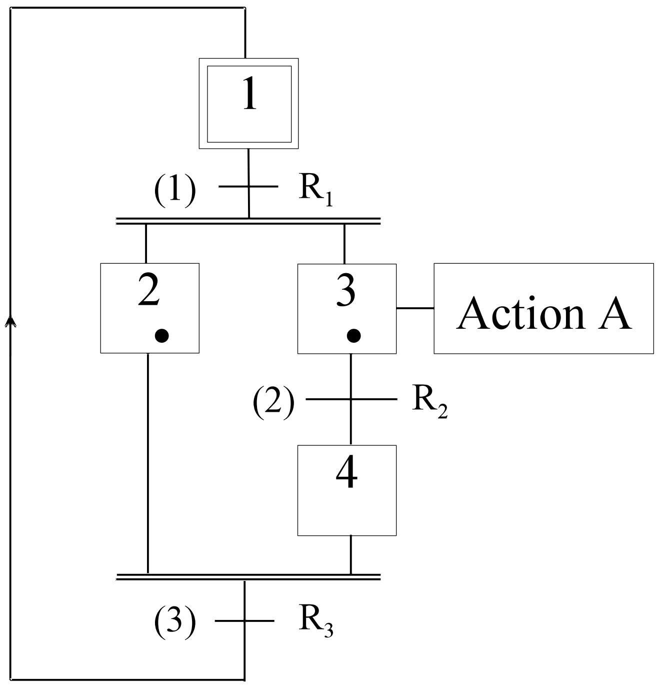
Basic Elements

Oriented connections (arcs)

In a GRAFCET:

An Arc can connect Steps to Transitions

An Arc can connect Transitions to Steps



A Step can have no Transitions as inputs (source);

A Step can have no Transitions as outputs (drain);

The same can occur for the Transitions.

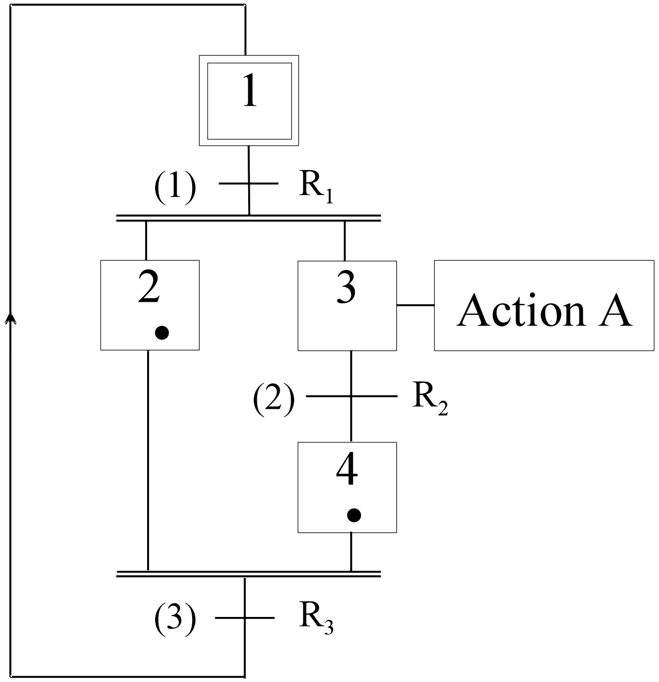
GRAFCET

State of a GRAFCET

The set of markings of a GRAFCET constitutes its state.

Question:

How evolves the state of a GRAFCET?



GRAFCET

State Evolution:

• Rule 1: Initial State

It is characterized by the active Steps at the beginning of operation (at least one).

• Rule 2: Transposition of a Transition

A Transition is active or enabled only if all the Steps at its input are active (if not it is inactive).

A Transition can only be transposed if it is active and its associated condition is true (receptivity function).

• Rule 3: Evolution of active Steps

The transposition of a Transition leads to the deactivation of all the Steps on its inputs and the activation of all Steps on its outputs.

• Rule 4: Simultaneous transposition of Transitions

All active Transitions are transposed simultaneously.

• Rule 5: Simultaneous activation and deactivation of a Step

In this case the activation has priority.

GRAFCET

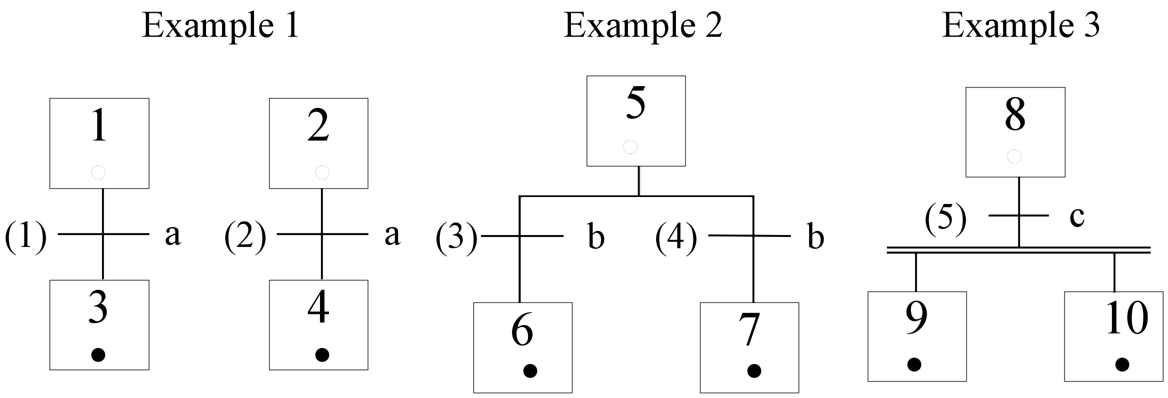
State Evolution:

- **Rule 2a:**

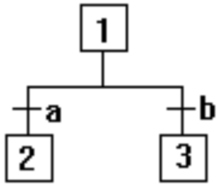
All active Transitions are transposed immediately.

- **Rule 4:**

Simultaneous active Transitions are transposed simultaneously.



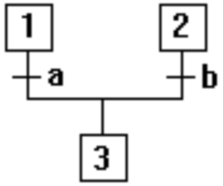
OR Divergences:



If Step 1 is active and if **a** is TRUE then Step 1 is deactivated and Step 2 is activated (state of Step 3 is maintained).

If **a** and **b** are TRUE and Step 1 is active then Step 1 is deactivated and Steps 2 and 3 are activated (for any previous state of Steps 2 and 3).

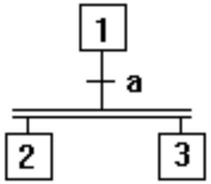
OR Convergences:



If Step 1 is active and if **a** is TRUE then Step 1 is deactivated and Step 3 is activated (state of Step 2 remains unchanged). The same happens for Step 2 and **b**.

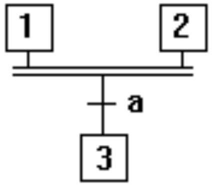
If both Steps 1 and 2 are active and **a** and **b** are TRUE then Steps 1 and 2 are deactivated and Step 3 is activated.

AND Divergences:



If Step 1 is active and if **a** is TRUE then Step 1 is deactivated and Steps 2 and 3 are activated.

AND Convergences:

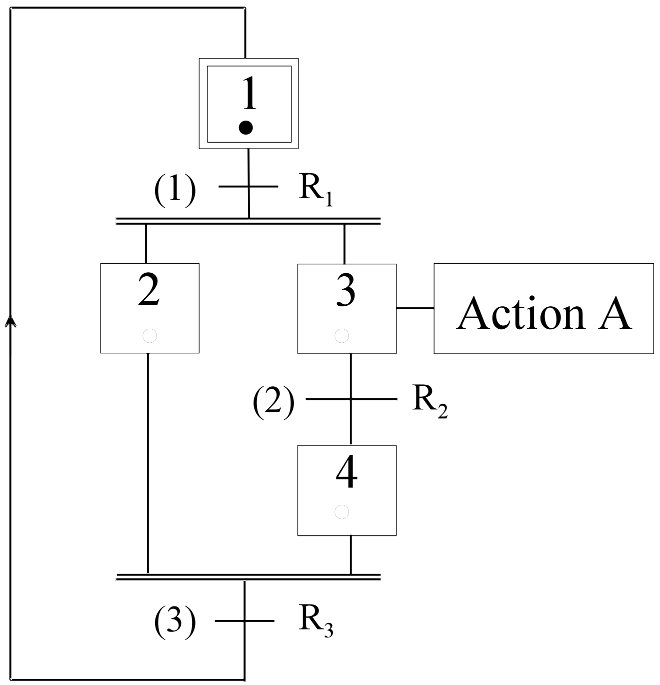


If Steps 1 and 2 are active and if **a** is TRUE then Steps 1 and 2 are deactivated and Step 3 is activated (if only one of the input steps is active, the state remains).

GRAFCET

Example:

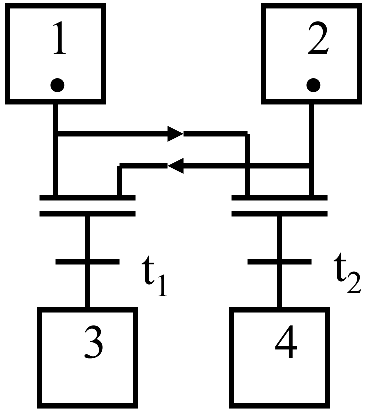
GRAFCET state evolution



Level activated Action. Actions can also be activated during transitions - see next.

GRAFCET

Modelling problem:



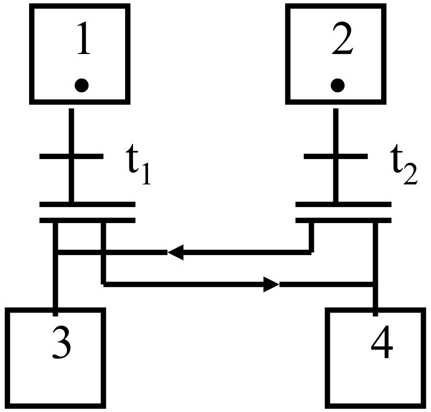
Given 4 Steps (1 to 4) and 2 Transitions (t1 and t2) write a segment of GRAFCET to solve the following problem:

In the case that the Steps 1 and 2 are active:

- if t1 is TRUE, activate Step 3 (and deactivate Steps 1 and 2);
- if t2 is TRUE, activate Step 4 (and deactivate Steps 1 and 2);
- otherwise, the state is maintained.

GRAFCET

Other modelling problem:



Given 4 Steps (1 to 4) and 2 Transitions (t1 and t2) write a segment of GRAFCET to solve the following problem:

If Step 1 is active and t1 is TRUE

OR

If Step 2 is active and t2 is TRUE

THEN

Activate Steps 3 and 4.

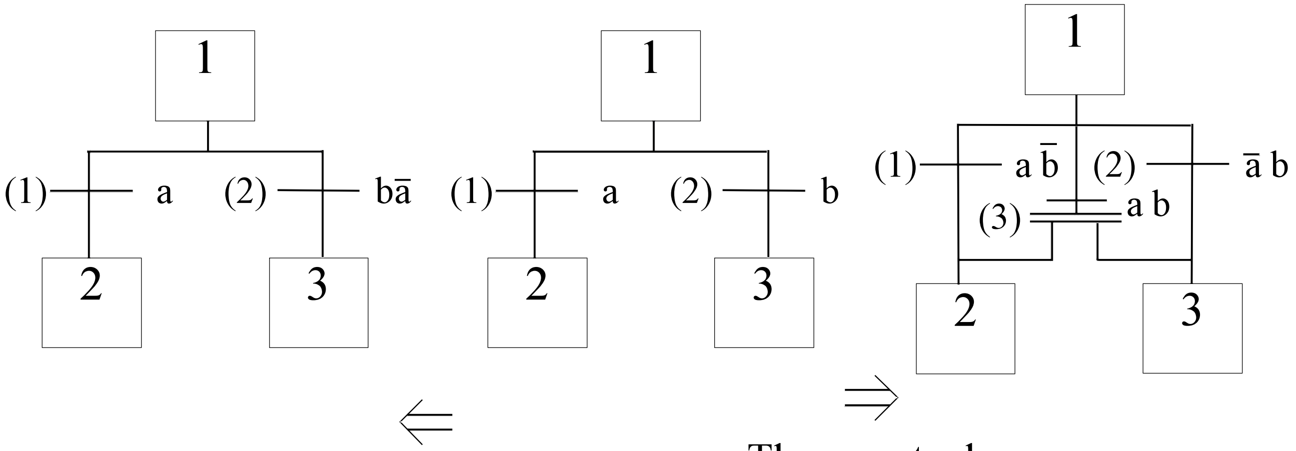
GRAFCET

GRAFCET state evolution:

Conflicts:

There exist Conflicts when the validation of a Transition depends on the same Step or when more than one receptivity functions can become true simultaneously.

Solutions:

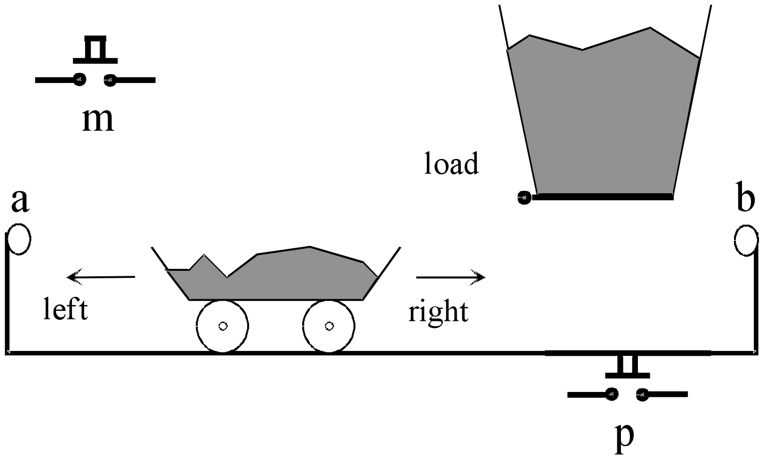


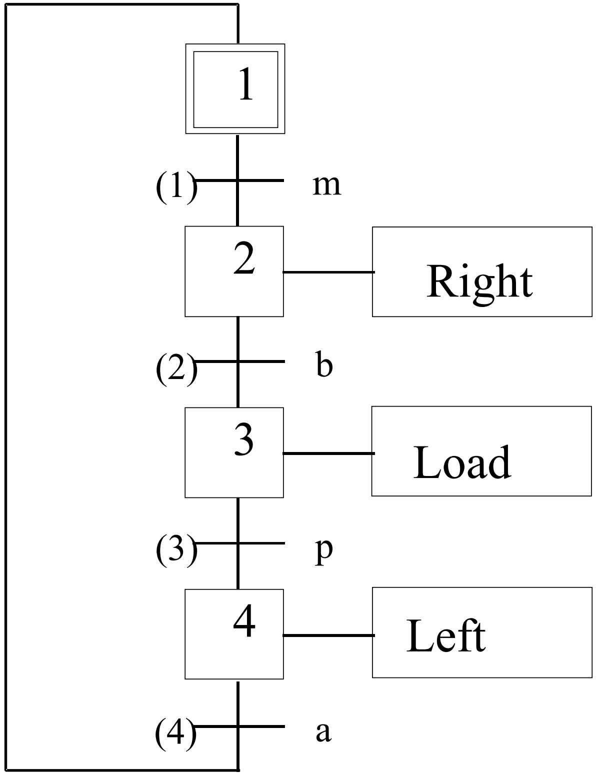
Transiction 1 prioritary

Three mutualy exclusive hypotheses

GRAFCET

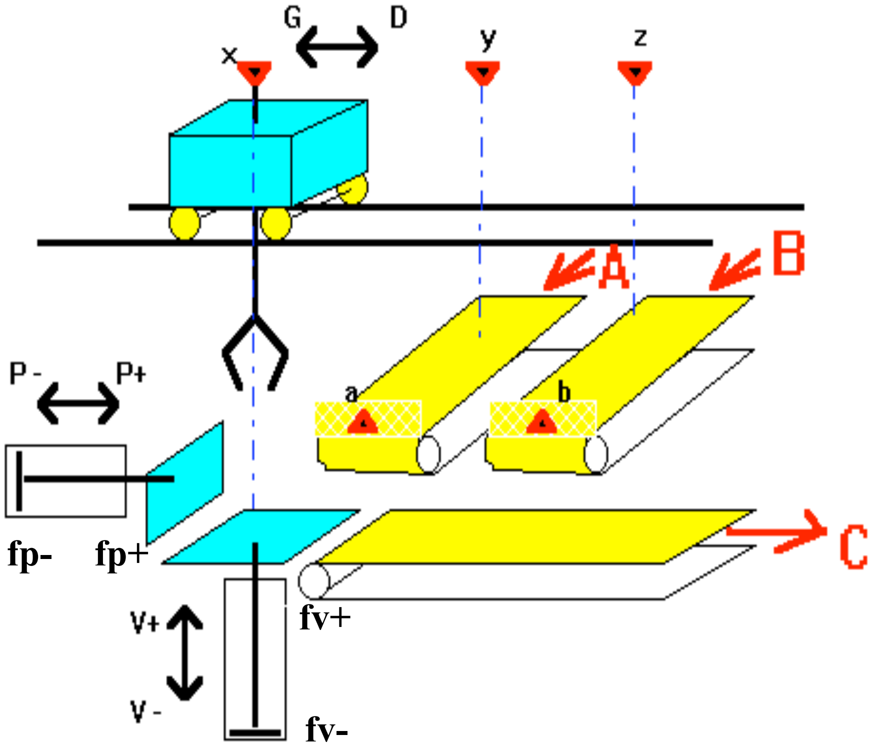
Example: modeling a control/automation system





GRAFCET

Example : modeling a automated transport workcell



* Conveyor **A,** with sensor **a** to detect the existence of part;

* Conveyor **B,** with sensor **b** to detect the existence of part;

•Manipulator on linear base commands **D** (droit) e **G (gauche)**, Sensors **x, y** e **z** to detect manipulator on base, over A, and over B, respectively.

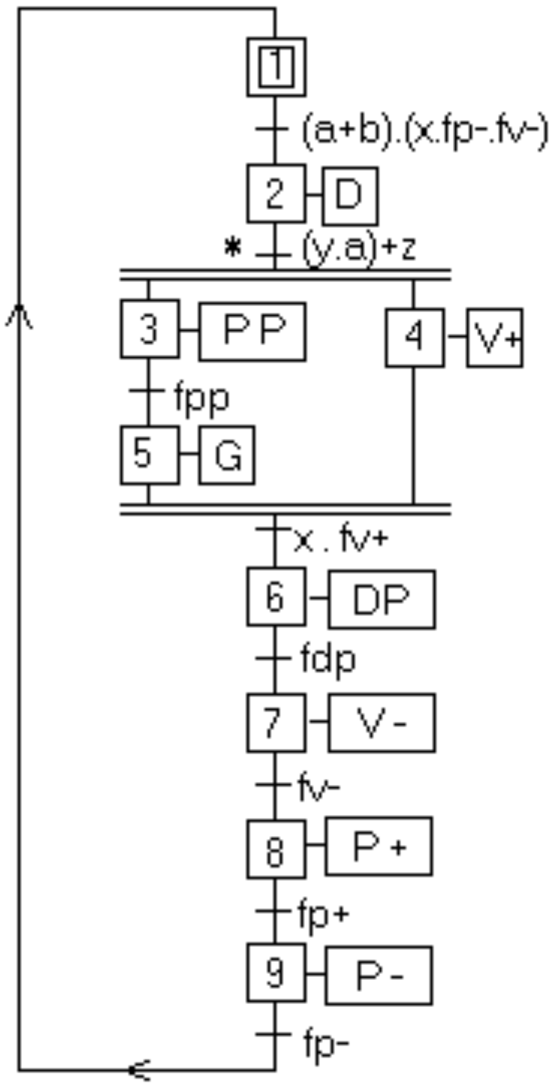
•Clamp with command to grab partt **PP**, and a limit switch (**fpp**). To unload part receives command **DP** and two limit Switches detect extremes of operation **fv+** on top and and **fv-** down.

* Efector to push parts with commands **P+** e **P-,** And two limit switches **fp+** e **fp-**.

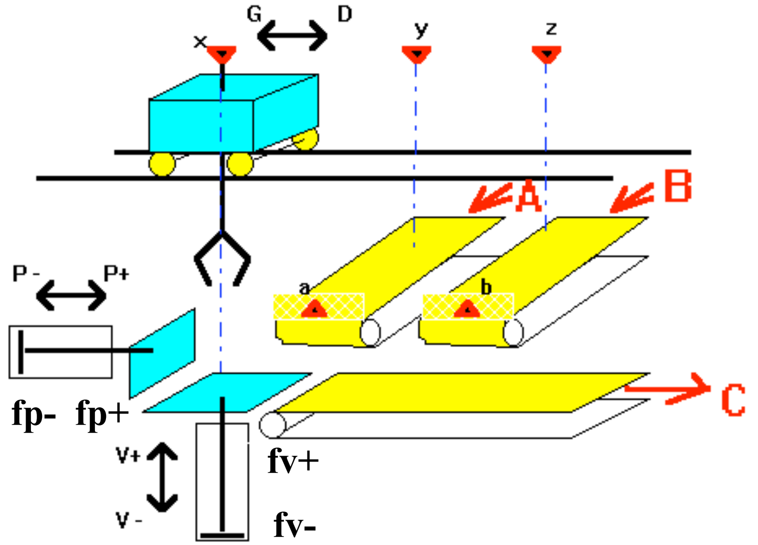
* The output conveyor is always ON.

*Conveyors **A** e **B** are commanded by other automata, independent of this workcell.

GRAFCET



Solution:

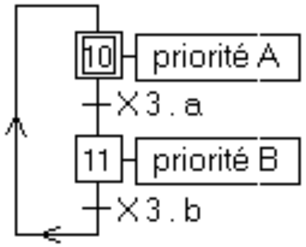


To guarantee alternate priorities, modify the programm with receptivity funcion (*)

$y.a.(\overline{b}+X10)+z$

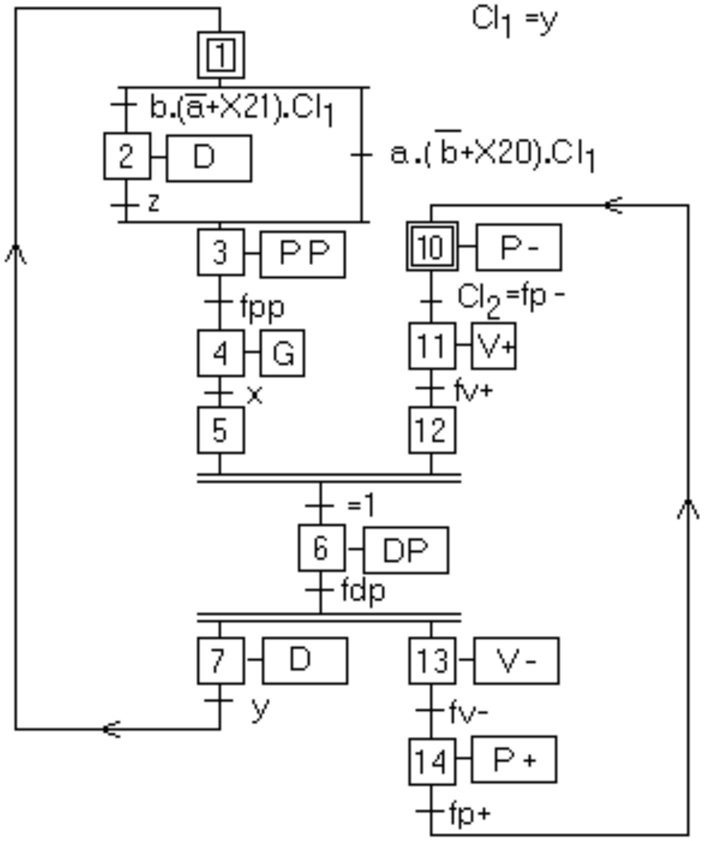
Meaning: grag part in **y,** if there exists part in **a** and if b is not prioritary; otherwise continue, stopping **b**.

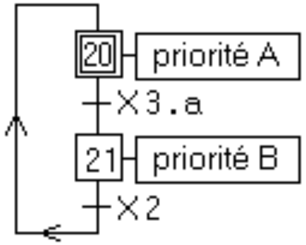
To implement the priorities add the following GRAFCET:

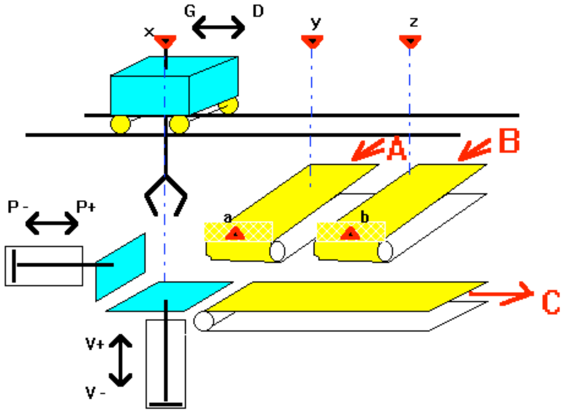


GRAFCET

Improved solution:







a)After part is processed search next;

b) Optimize the base of the manipulator to reduce delays –

obvious solution: **y**.

GRAFCET

Example: modeling and automation of a distribution system

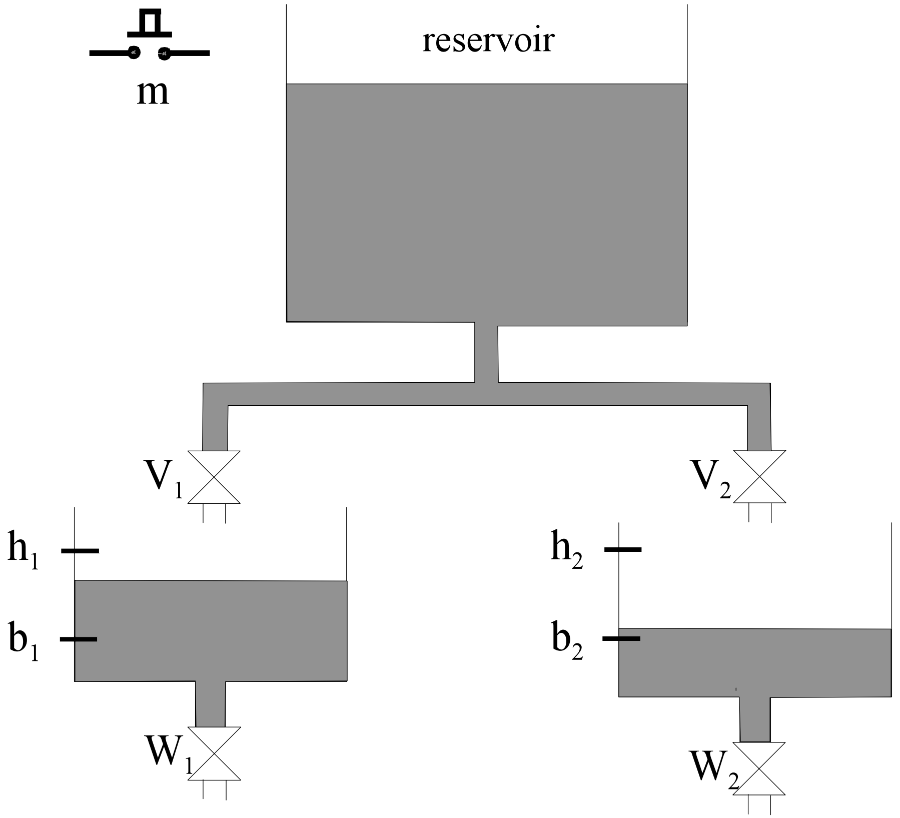
Sensors:

m

b_1, h_1, b_2 e h_2

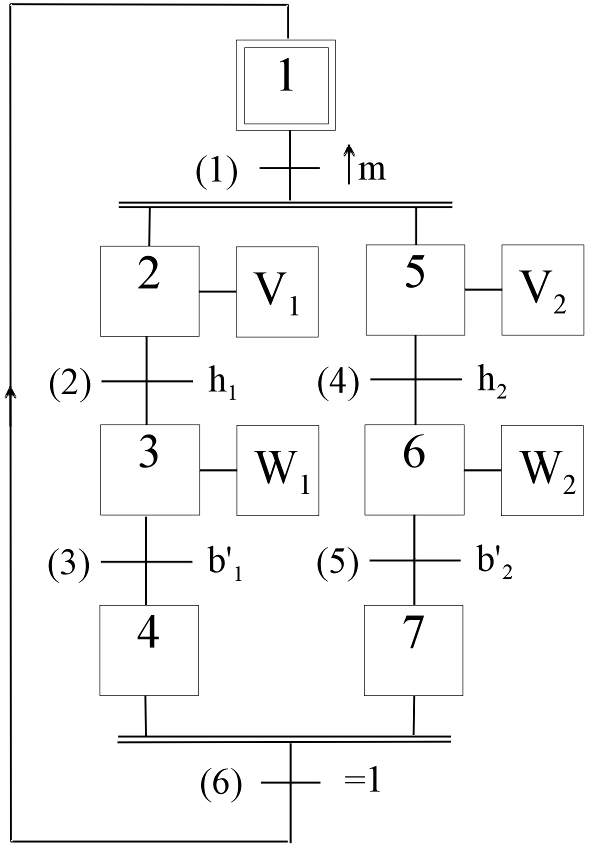
Actuators:

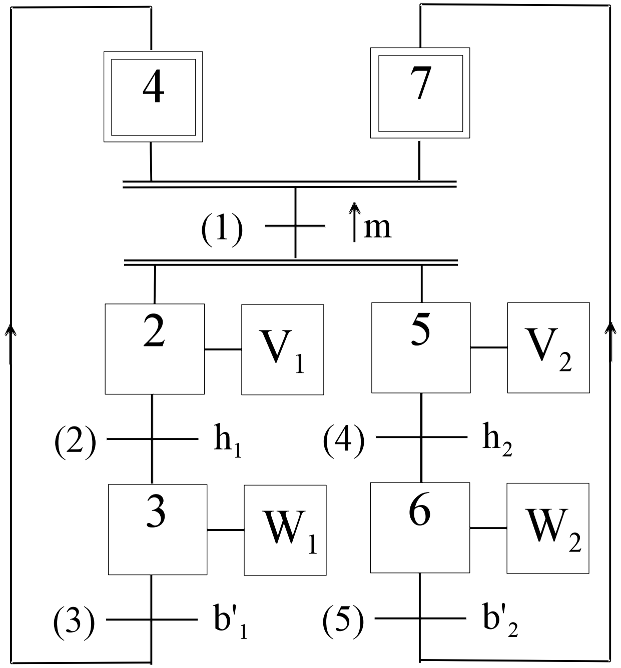
V_1, V_2, W_1 e W_2



GRAFCET

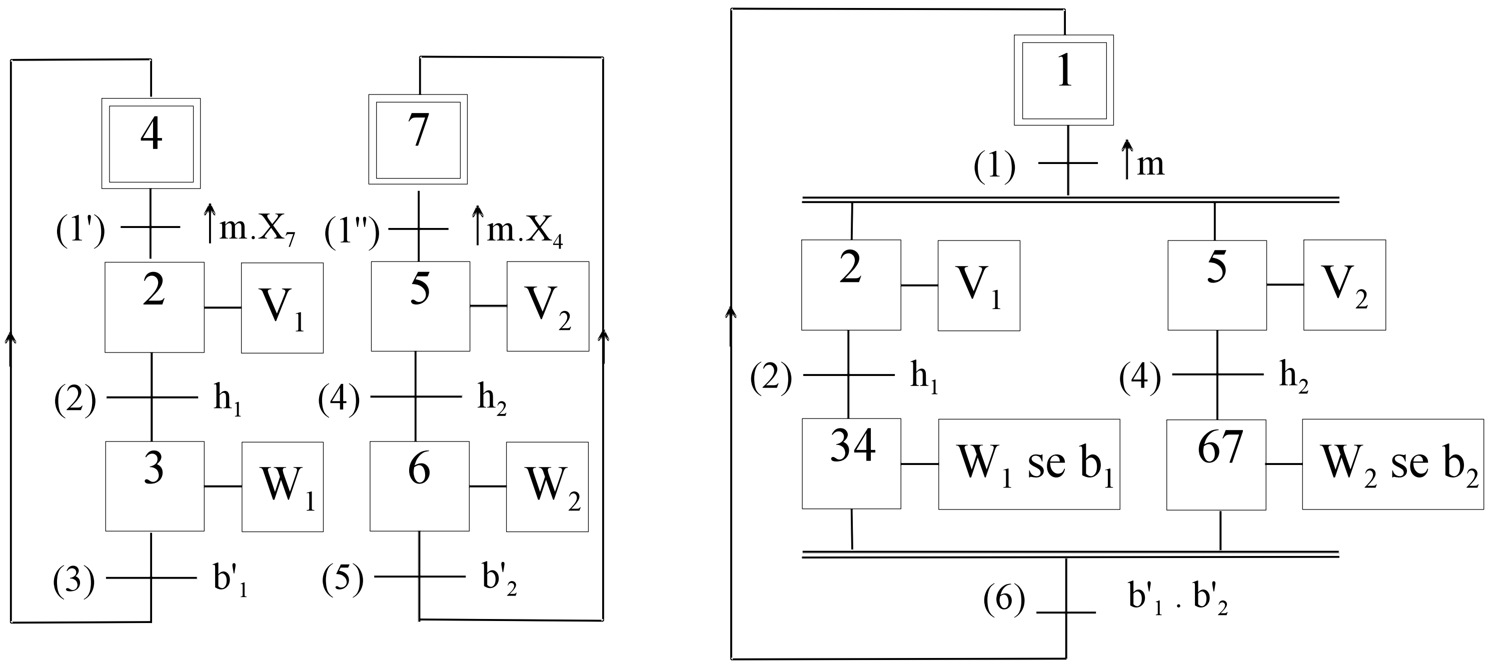
Example: modeling and automation of a distribution system





GRAFCET

Example: modeling and automation of a distribution system



GRAFCET

Events: Properties

$\uparrow a = \downarrow a'$

$\uparrow a \,.\, a = \uparrow a$ $\quad$ $\uparrow a \,.\, a' = 0$ $\quad$ $\downarrow a \,.\, a' = \downarrow a$ $\quad$ $\downarrow a \,.\, a = 0$

$\uparrow a \,.\, \uparrow a = \uparrow a$ $\quad$ $\uparrow a \,.\, \uparrow a' = 0$

$\uparrow (a \,.\, b) = \uparrow a \,.\, b + \uparrow b \,.\, a$ $\quad$ $\uparrow (a + b) = \uparrow a \,.\, b' + \uparrow b \,.\, a'$

$\uparrow (a \,.\, b) \,.\, \uparrow (a \,.\, c) = \uparrow \quad (a \,.\, b \,.\, c)$

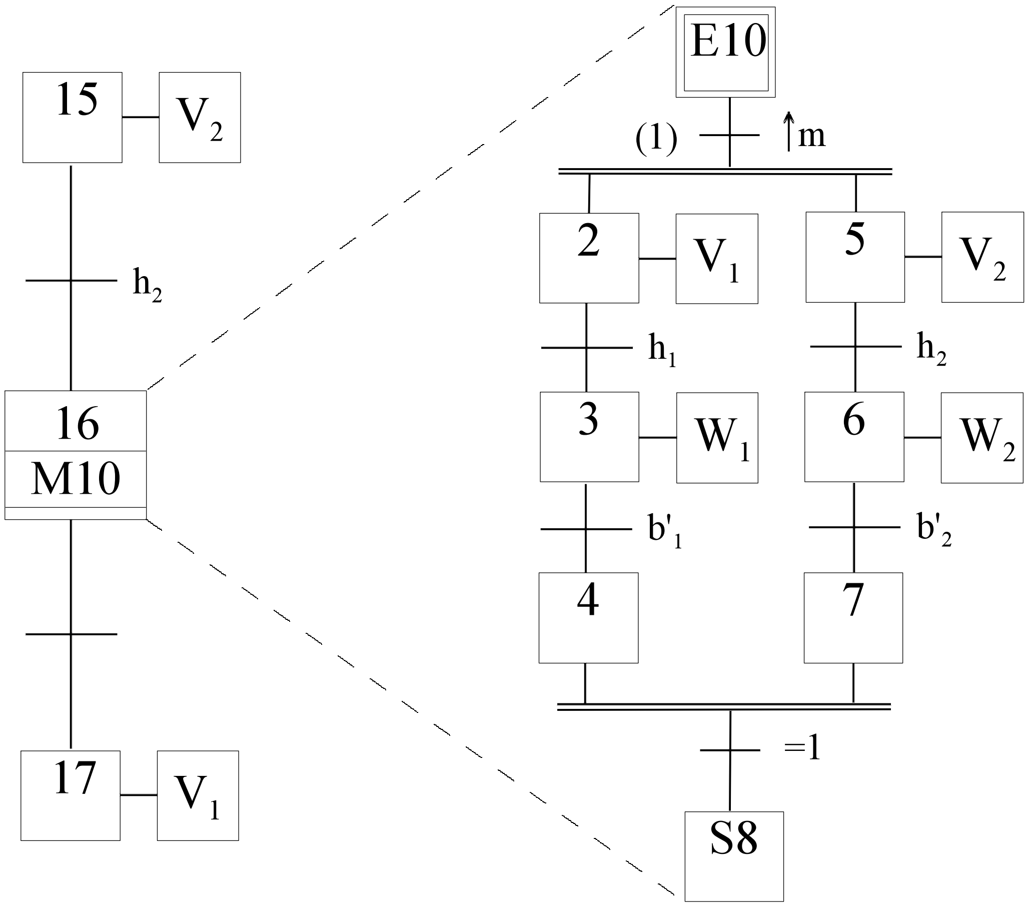
in general, if events a and b are independents

$\uparrow a \,.\, \uparrow b = 0$

GRAFCET

Other auxiliary mechanisms

Macro-steps



GRAFCET

Other auxiliary mechanisms

Pseudo Macro-steps

Macro Actions

- **Force actions**
- **Enable actions**
- **Mask actions**

GRAFCET

Implementation in DOLOG80

The activity of each Step is stored in an auxiliary memory.

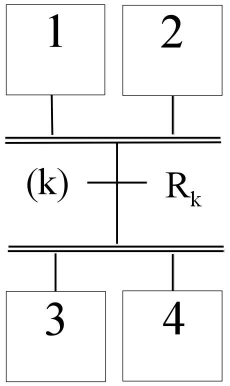
At startup do:

```
AM128
SLMx
...
AM128
SLMy
 (initial steps)
RLM128
```

Store R_k evaluation in M100

```
AM1
AM2
AM100
SLM3
AM1
AM2
AM100
SLM4
```

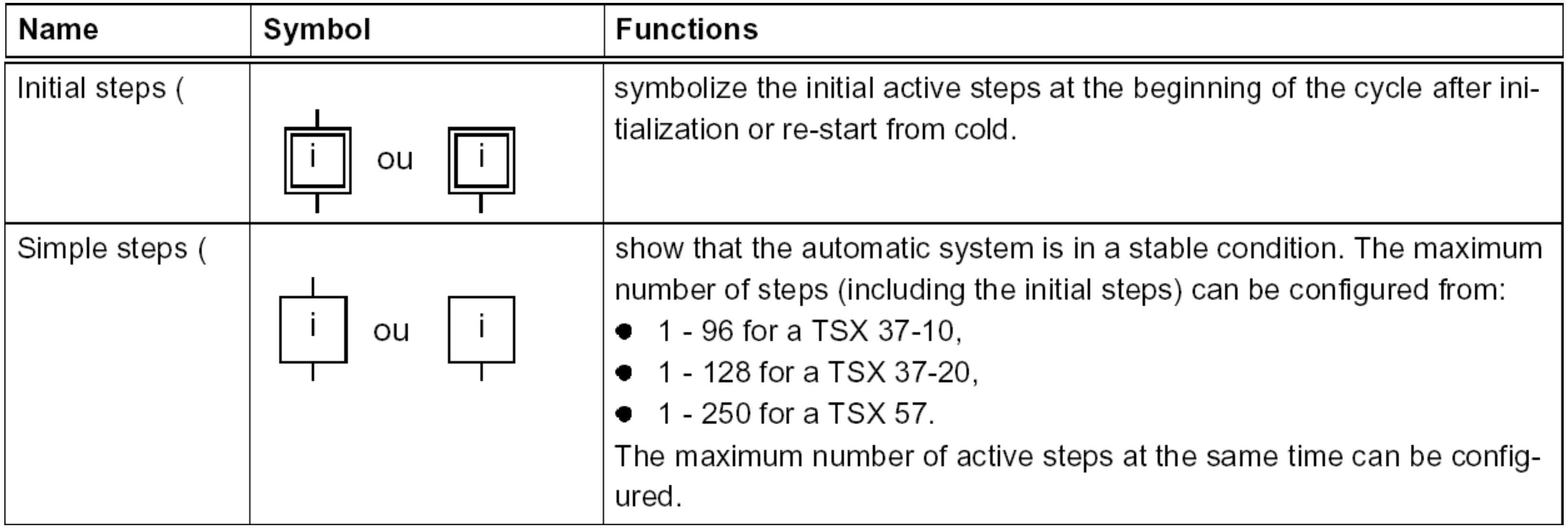
```
AM3
AM4
RLM1
AM3
AM4
RLM2
```



GRAFCET

Implementation in the TSX3722/TSX57

Steps

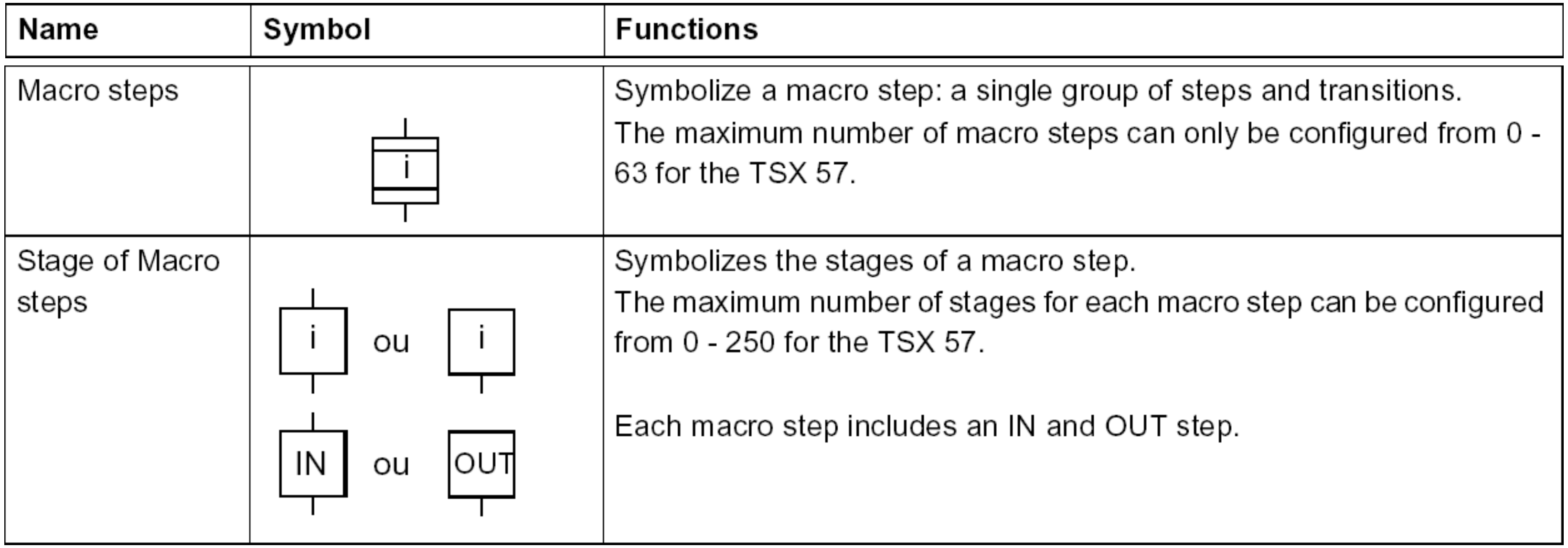


Name	Symbol	Functions
Initial steps (	i ou i	symbolize the initial active steps at the beginning of the cycle after initialization or re-start from cold.
Simple steps (	i ou i	show that the automatic system is in a stable condition. The maximum number of steps (including the initial steps) can be configured from: ● 1 - 96 for a TSX 37-10, ● 1 - 128 for a TSX 37-20, ● 1 - 250 for a TSX 57. The maximum number of active steps at the same time can be configured.

GRAFCET

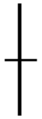
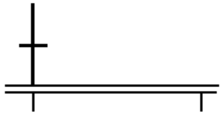
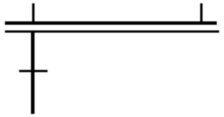
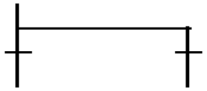
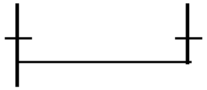
Implementation in the TSX3722/TSX57

Macro-steps



Name	Symbol	Functions
Macro steps	i	Symbolize a macro step: a single group of steps and transitions. The maximum number of macro steps can only be configured from 0 - 63 for the TSX 57.
Stage of Macro steps	i ou i IN ou OUT	Symbolizes the stages of a macro step. The maximum number of stages for each macro step can be configured from 0 - 250 for the TSX 57. Each macro step includes an IN and OUT step.

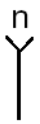
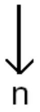
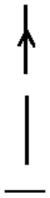
GRAFCET

Name	Symbol	Functions
Transitions		allow the transfer from one step to another. A transition condition associated with this condition is used to define the logic conditions necessary to cross this transition. The maximum number of transitions is 1024. It cannot be configured. The maximum number of valid transitions at the same time can be configured.
AND divergences		Transition from one step to several steps: is used to activate a maximum of 11 steps at the same time.
AND convergences		Transition of several steps to one: is used to deactivate a maximum of 11 steps at the same time.
OR divergences		Transition from one step to several steps: is used to carry out a switch to a maximum of 11 steps.
OR convergences		Transition of several steps to one: is used to end switching from a maximum of 11 steps.

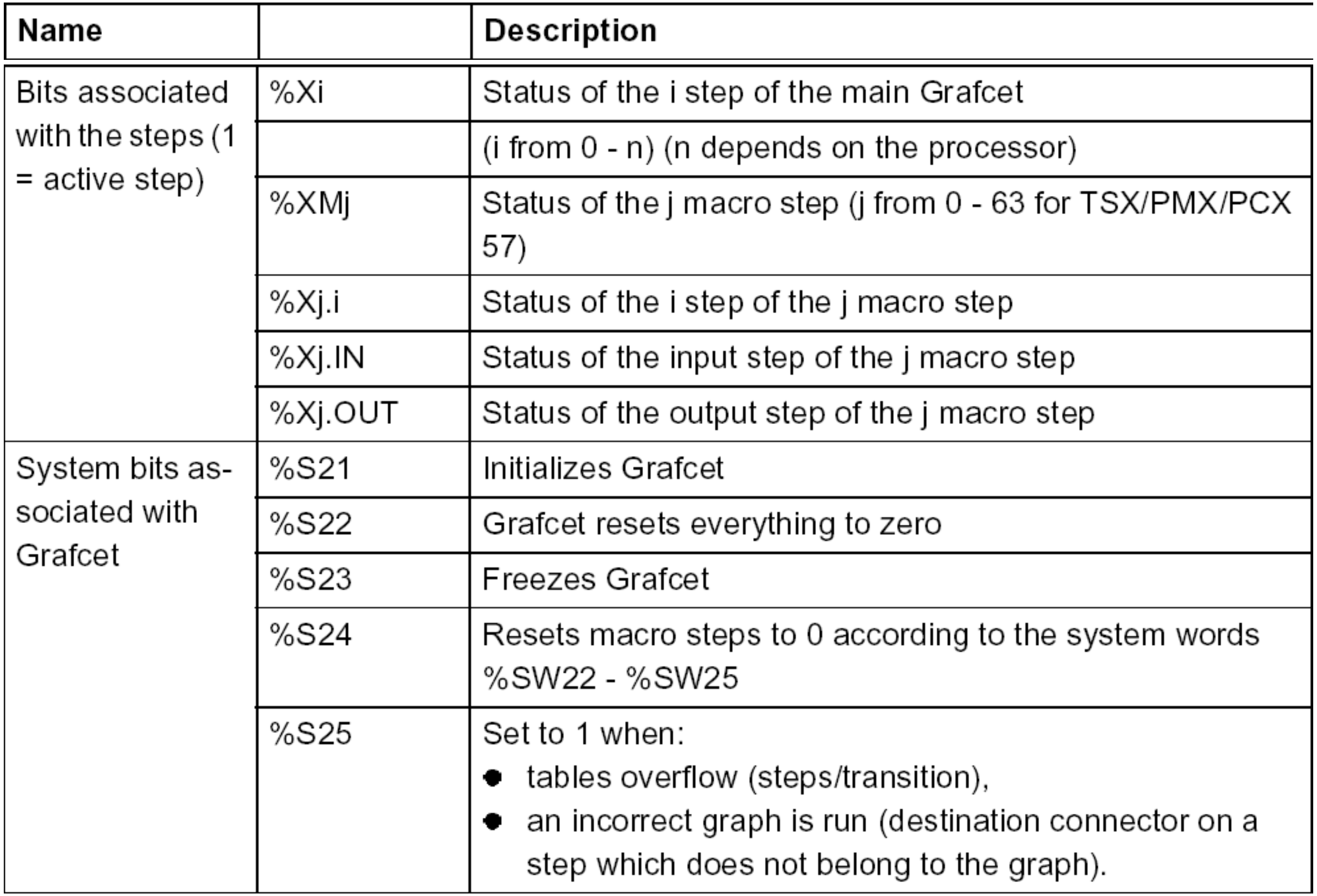
GRAFCET

Implementation in the TSX3722/TSX57

Arcs/Connectors

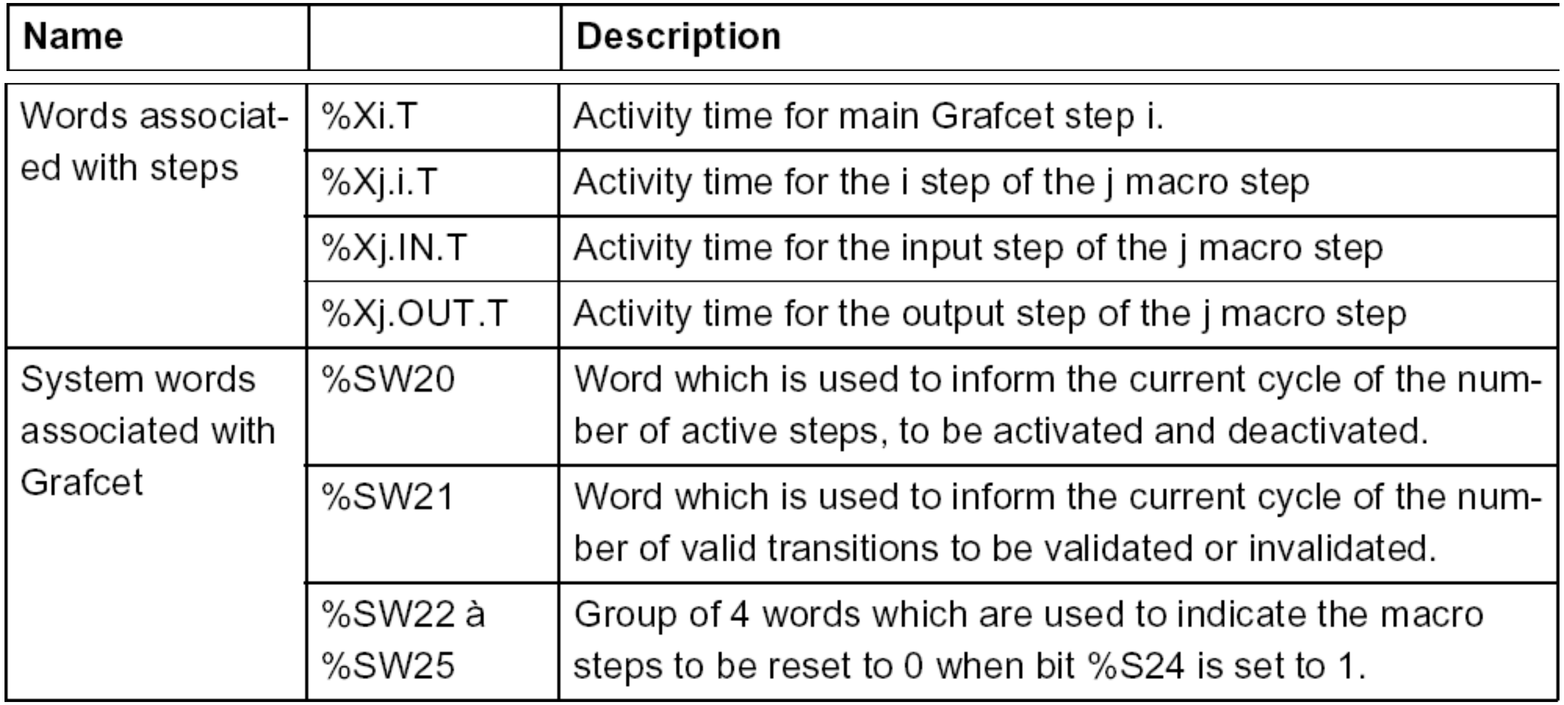
Name	Symbol	Functions
Source connectors		"n" is the number of the step "it comes from" (source step).
Destination connector		"n" is the number of the step "it's going to" (target step).
Links directed towards: ● top ● bottom ● right or left		These links are used for switching, jumping a step, restarting steps (sequence).

Information associated with Steps in the GRAFCET:



Name		Description
Bits associated with the steps (1 = active step)	%Xi	Status of the i step of the main Grafcet
		(i from 0 - n) (n depends on the processor)
	%XMj	Status of the j macro step (j from 0 - 63 for TSX/PMX/PCX 57)
	%Xj.i	Status of the i step of the j macro step
	%Xj.IN	Status of the input step of the j macro step
	%Xj.OUT	Status of the output step of the j macro step
System bits associated with Grafcet	%S21	Initializes Grafcet
	%S22	Grafcet resets everything to zero
	%S23	Freezes Grafcet
	%S24	Resets macro steps to 0 according to the system words %SW22 - %SW25
	%S25	Set to 1 when: ● tables overflow (steps/transition), ● an incorrect graph is run (destination connector on a step which does not belong to the graph).

Information associated with Steps in the GRAFCET (bis):



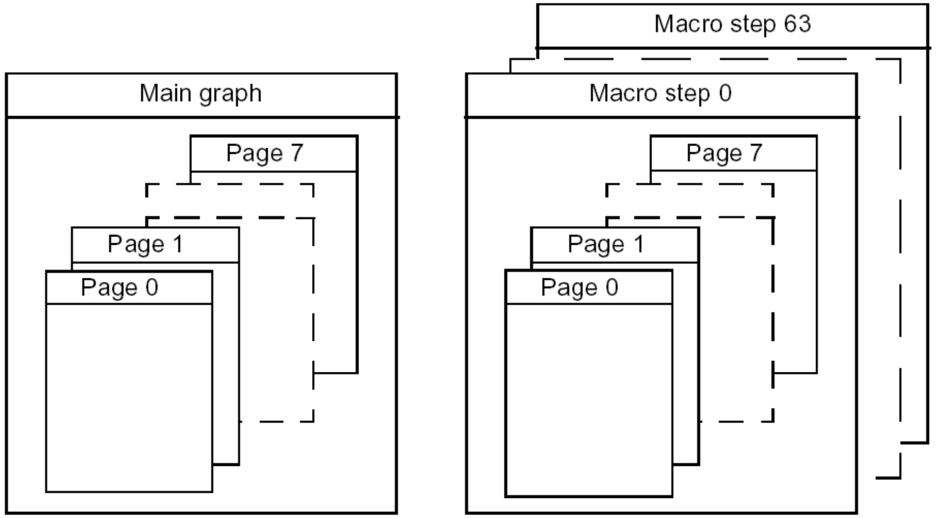
Name		Description
Words associated with steps	%Xi.T	Activity time for main Grafcet step i.
	%Xj.i.T	Activity time for the i step of the j macro step
	%Xj.IN.T	Activity time for the input step of the j macro step
	%Xj.OUT.T	Activity time for the output step of the j macro step
System words associated with Grafcet	%SW20	Word which is used to inform the current cycle of the number of active steps, to be activated and deactivated.
	%SW21	Word which is used to inform the current cycle of the number of valid transitions to be validated or invalidated.
	%SW22 à %SW25	Group of 4 words which are used to indicate the macro steps to be reset to 0 when bit %S24 is set to 1.

And where to find information related with Transitions?

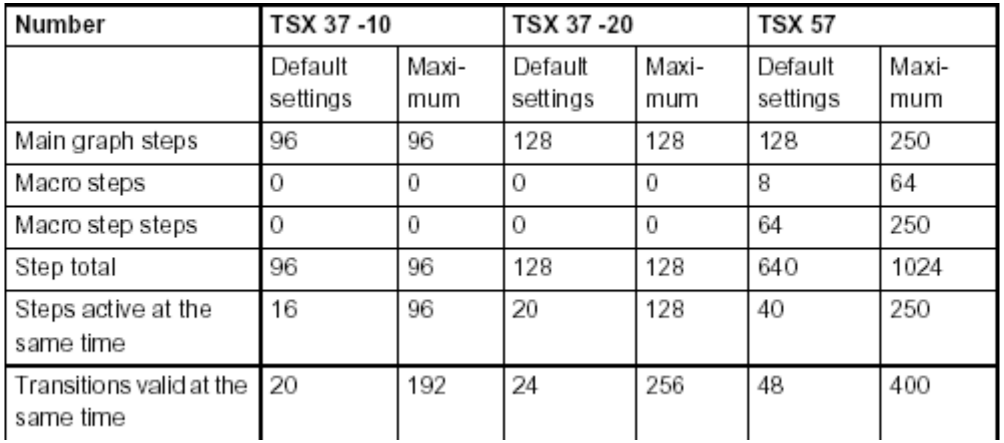
Does not make sense state or activity nor timmings
(only number of occurences).

GRAFCET

General structure:



Characteristics:



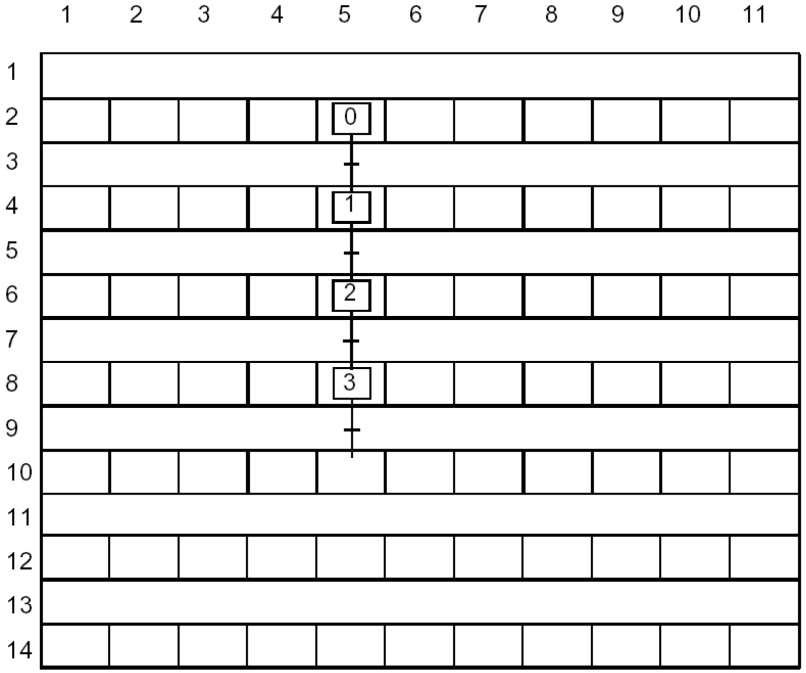
Number	TSX 37 -10		TSX 37 -20		TSX 57	
	Default settings	Maximum	Default settings	Maximum	Default settings	Maximum
Main graph steps	96	96	128	128	128	250
Macro steps	0	0	0	0	8	64
Macro step steps	0	0	0	0	64	250
Step total	96	96	128	128	640	1024
Steps active at the same time	16	96	20	128	40	250
Transitions valid at the same time	20	192	24	256	48	400

GRAFCET

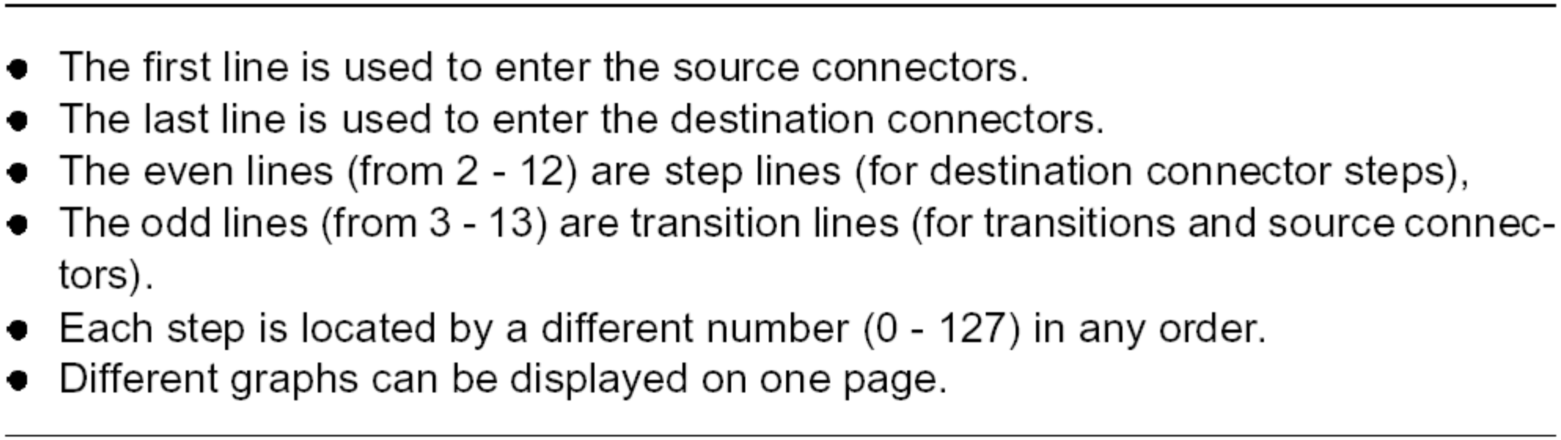
Editor: 8 páginas

- Pages 0 to 7

- 154 cells (14*11)



Characteristics:

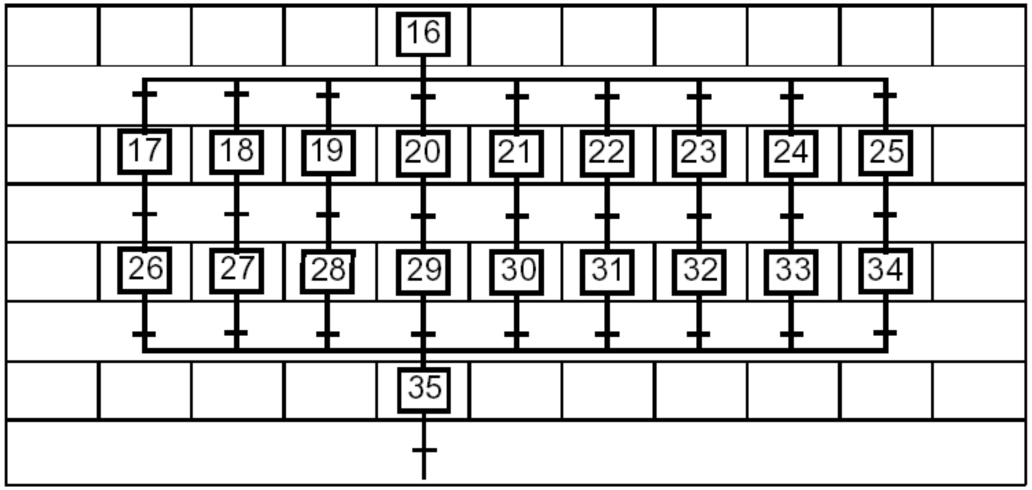


- The first line is used to enter the source connectors.
- The last line is used to enter the destination connectors.
- The even lines (from 2 - 12) are step lines (for destination connector steps),
- The odd lines (from 3 - 13) are transition lines (for transitions and source connectors).
- Each step is located by a different number (0 - 127) in any order.
- Different graphs can be displayed on one page.

GRAFCET

OR divergences

(OR convergences)



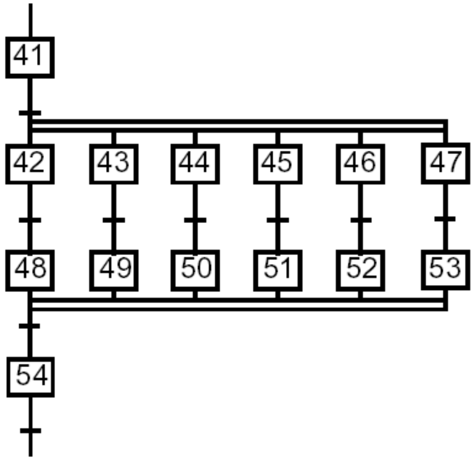
Characteristics:

- The number of transitions upstream of a switching end (OR convergence) or downstream of a switching (OR divergence) must not exceed 11.
- Switching can be to the left or to the right.
- Switching must general finish with switching end.
- To avoid crossing several transitions at the same time, the associated transition conditions must be exclusive.

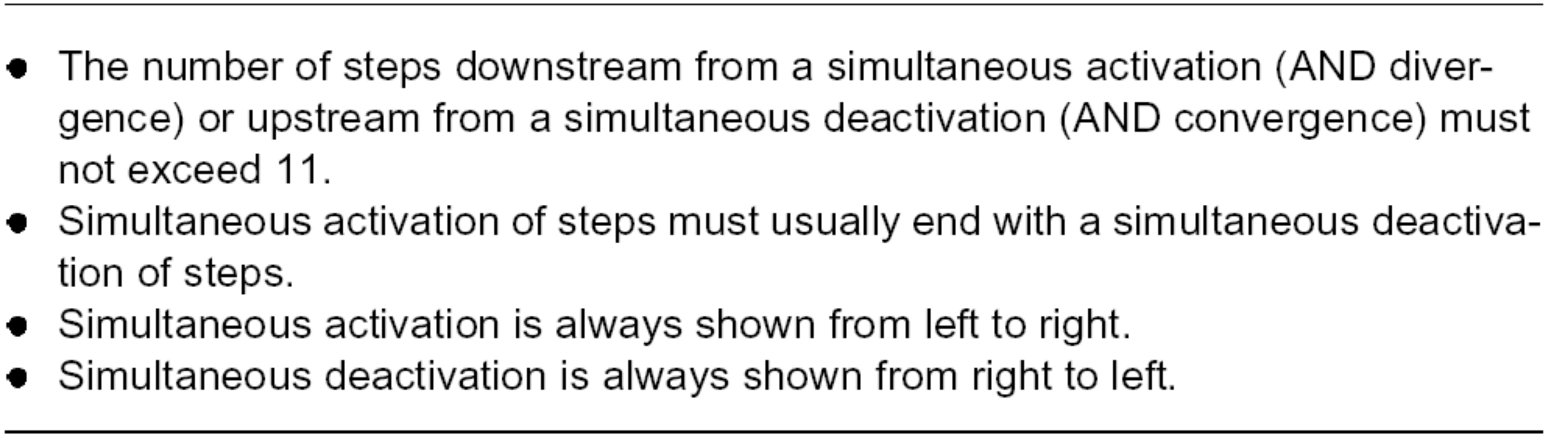
GRAFCET

AND divergences

(AND Convergences)



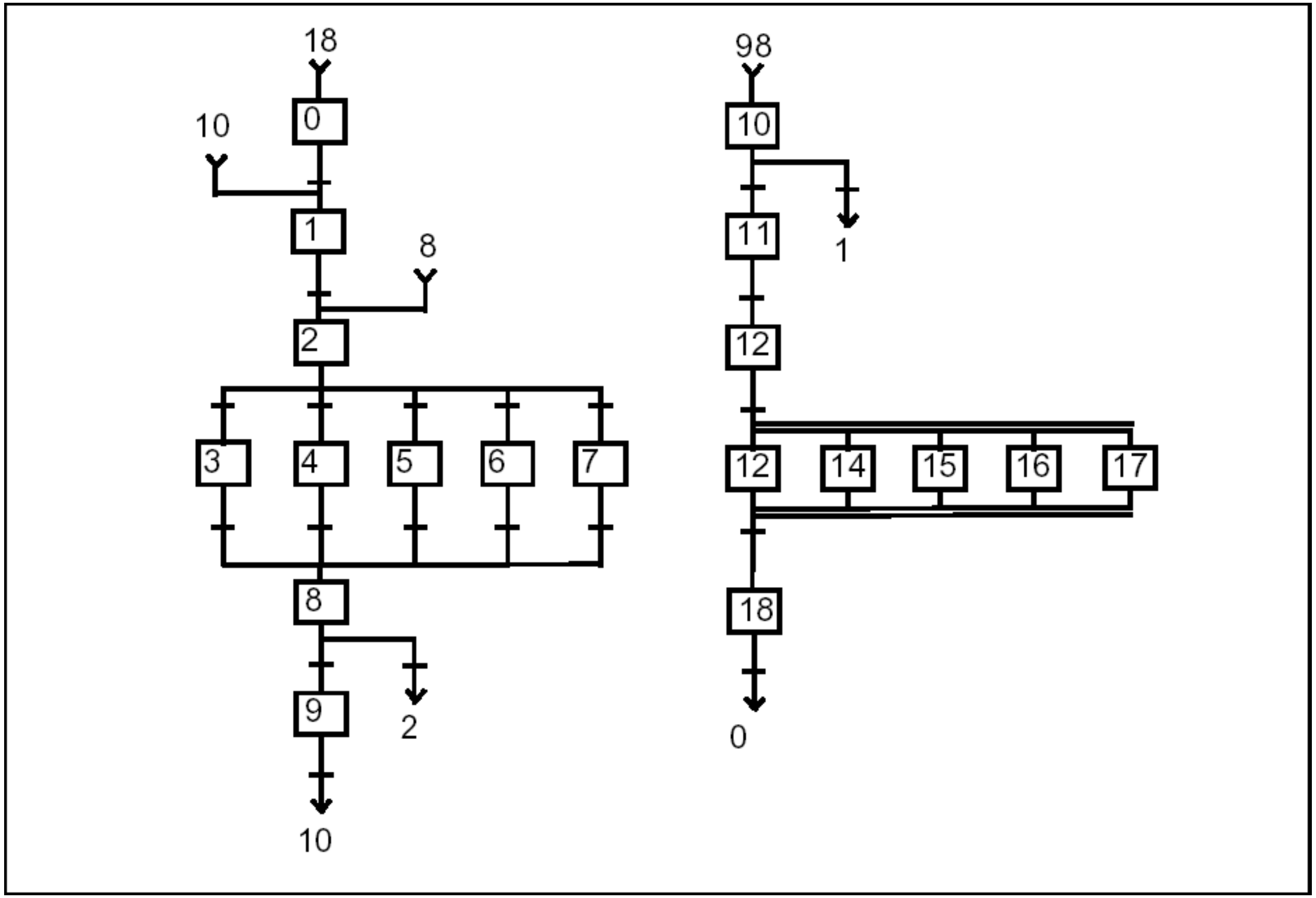
Characteristics:



- The number of steps downstream from a simultaneous activation (AND divergence) or upstream from a simultaneous deactivation (AND convergence) must not exceed 11.
- Simultaneous activation of steps must usually end with a simultaneous deactivation of steps.
- Simultaneous activation is always shown from left to right.
- Simultaneous deactivation is always shown from right to left.

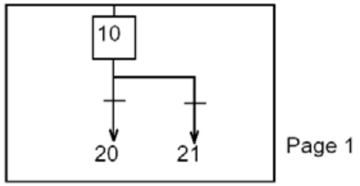
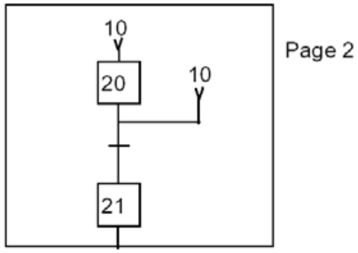
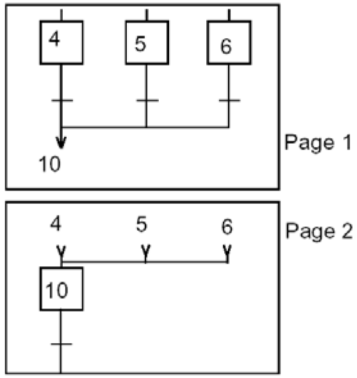
GRAFCET

Arcs/Connectors

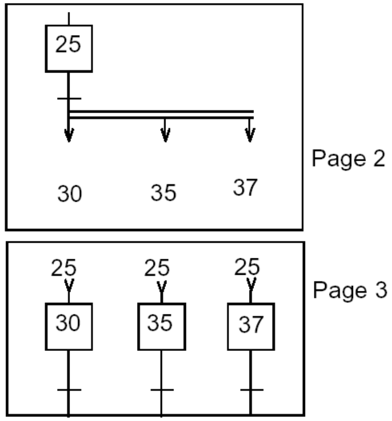
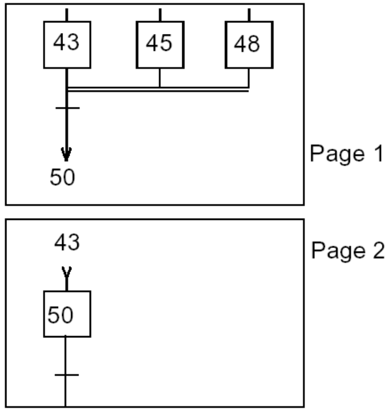


Rules for divergences and convergences:

OR

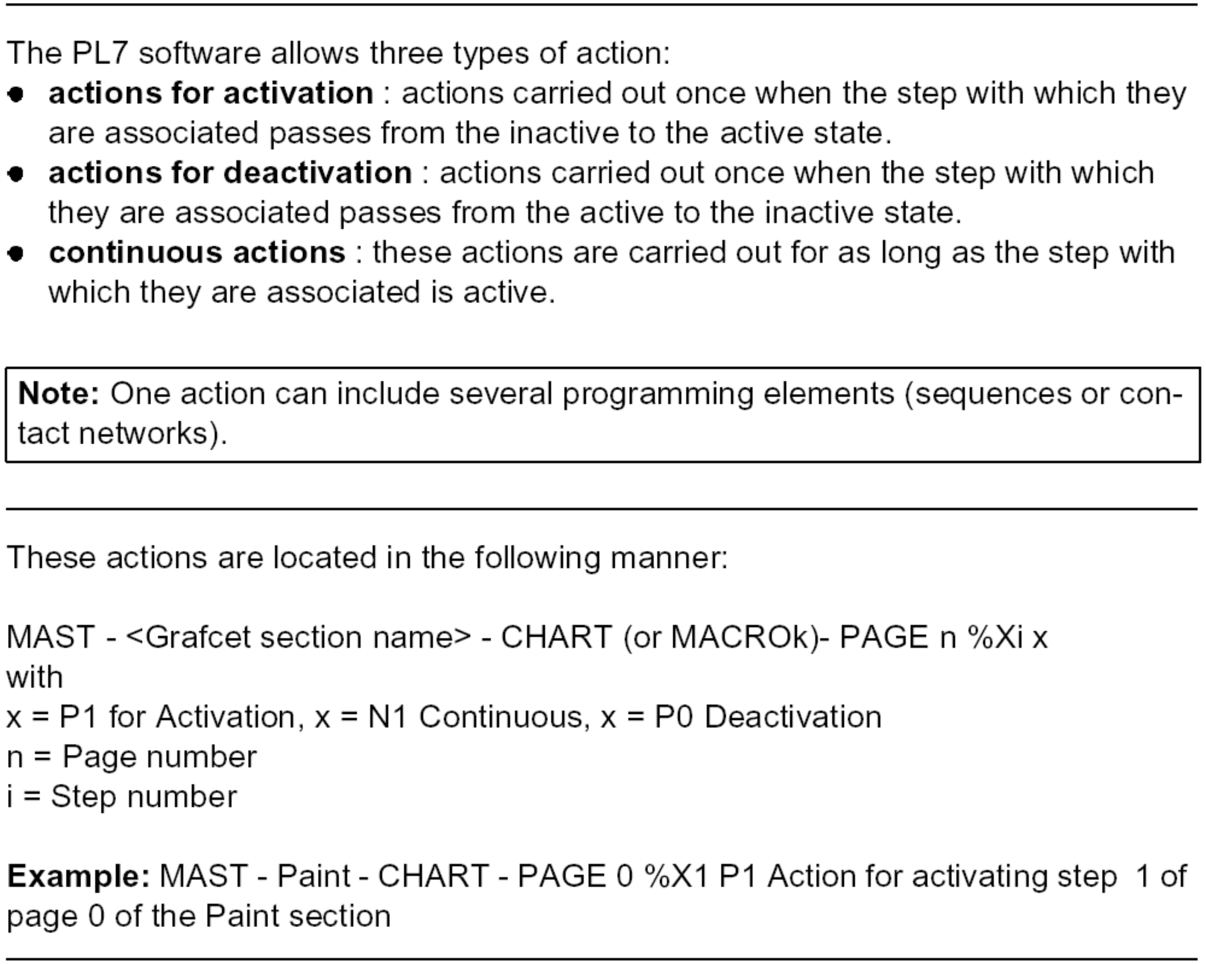
Rule	Illustration
For switching, transitions and destination connectors must be entered on the same page.	
To end switching, the source connectors must be entered on the same page as the destination step.	
For an end to switching followed by a return to destination, there must be as many source connectors as steps before the end of switching.	

AND

Rule	Illustration
To activate steps simultaneously, the destination connectors must be on the same page as the divergence step and transition.	 25 30 35 37 Page 2 25 25 25 30 35 37 Page 3
To deactivate simultaneously, the convergence steps and transition must be on the same page as the destination connector. When several steps converge onto one transition, the source connector has the number of the furthest upstream step on the left.	

GRAFCET

Programming Actions



The PL7 software allows three types of action:

- **actions for activation** : actions carried out once when the step with which they are associated passes from the inactive to the active state.
- **actions for deactivation** : actions carried out once when the step with which they are associated passes from the active to the inactive state.
- **continuous actions** : these actions are carried out for as long as the step with which they are associated is active.

Note: One action can include several programming elements (sequences or contact networks).

These actions are located in the following manner:

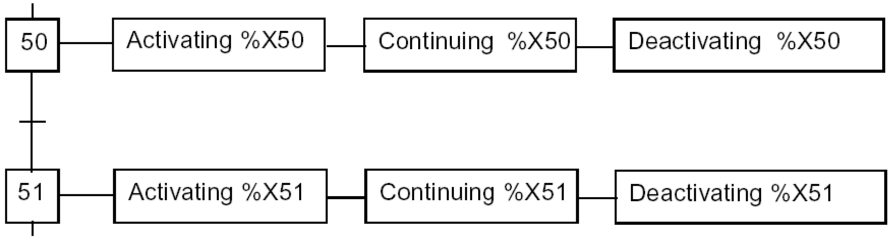
MAST - <Grafcet section name> - CHART (or MACROk)- PAGE n %Xi x
with
x = P1 for Activation, x = N1 Continuous, x = P0 Deactivation
n = Page number
i = Step number

Example: MAST - Paint - CHART - PAGE 0 %X1 P1 Action for activating step 1 of page 0 of the Paint section

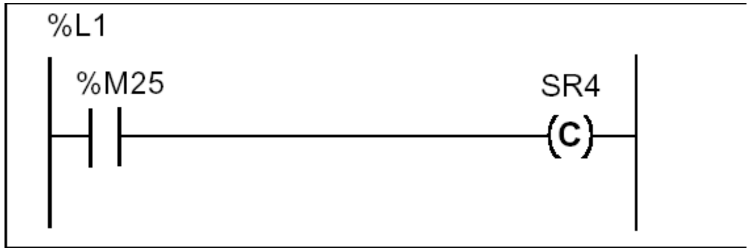
GRAFCET

Programming Actions

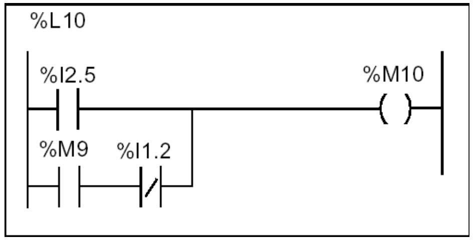
Example of execution of Actions



Example of Activation/deactivation

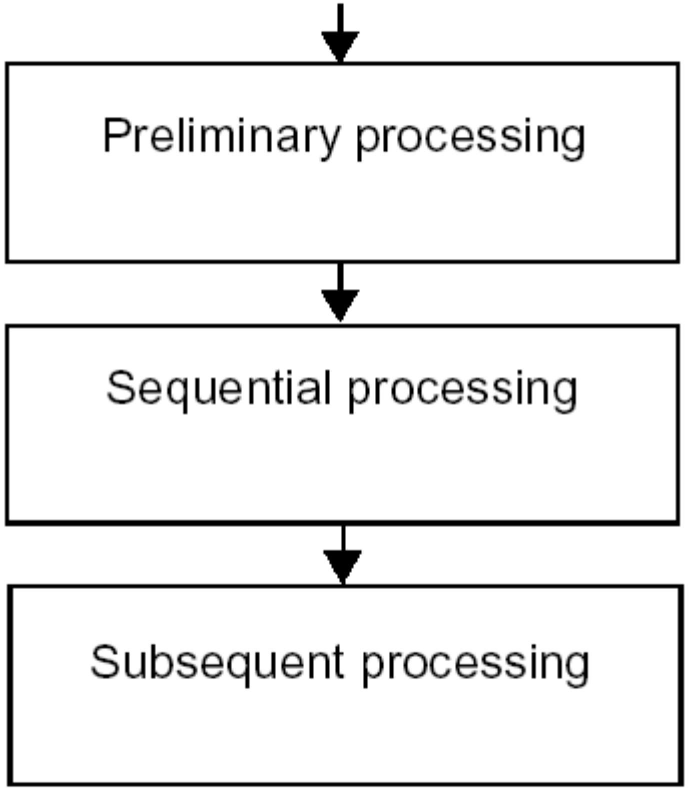


Example of continuous Action



GRAFCET

GRAFCET Section Sctructure



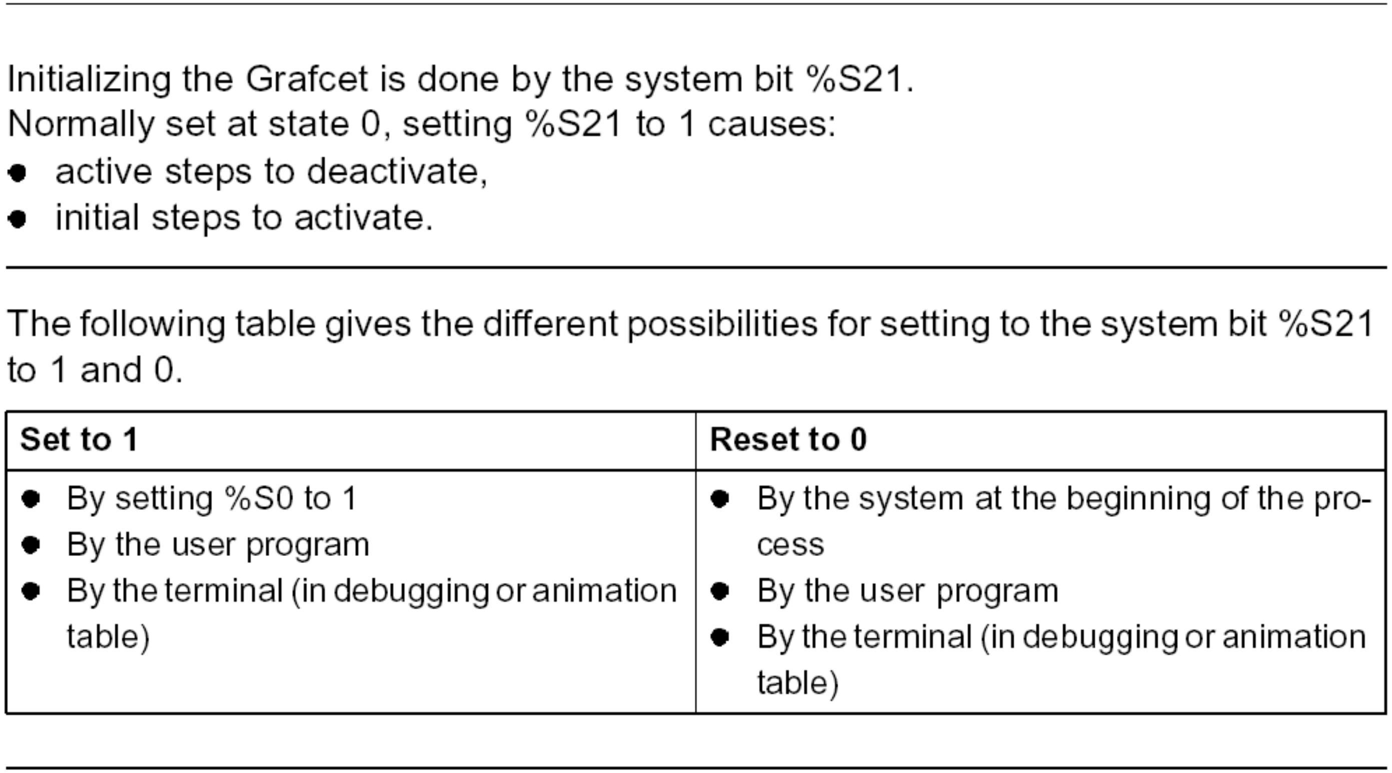
LD, IL, ST

GRAFCET

LD, IL, ST

GRAFCET

GRAFCET Section Initialization



Initializing the Grafcet is done by the system bit %S21.
Normally set at state 0, setting %S21 to 1 causes:

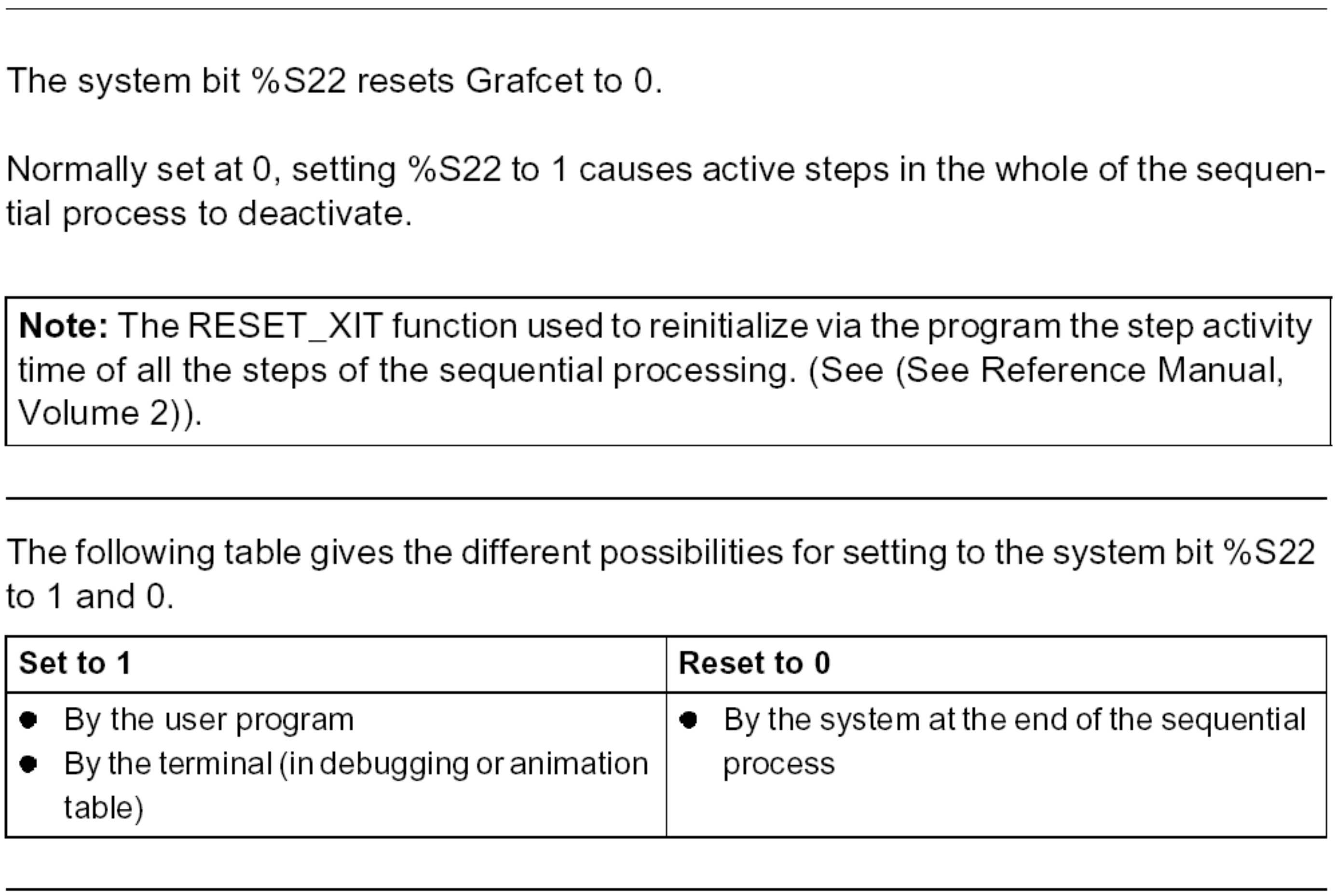
- active steps to deactivate,
- initial steps to activate.

The following table gives the different possibilities for setting to the system bit %S21 to 1 and 0.

Set to 1	Reset to 0
● By setting %S0 to 1 ● By the user program ● By the terminal (in debugging or animation table)	● By the system at the beginning of the process ● By the user program ● By the terminal (in debugging or animation table)

GRAFCET

GRAFCET Section Reset



The system bit %S22 resets Grafcet to 0.

Normally set at 0, setting %S22 to 1 causes active steps in the whole of the sequential process to deactivate.

Note: The RESET_XIT function used to reinitialize via the program the step activity time of all the steps of the sequential processing. (See (See Reference Manual, Volume 2)).

The following table gives the different possibilities for setting to the system bit %S22 to 1 and 0.

Set to 1	Reset to 0
● By the user program ● By the terminal (in debugging or animation table)	● By the system at the end of the sequential process

Industrial Automation

(Automação de Processos Industriais)

CAD/CAM and CNC

http://www.isr.ist.utl.pt/~pjcro/courses/api0809/api0809.html

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

Syllabus:

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

...

Chap. 5 – CAD/CAM and CNC [1 semana]

Methodology CAD/CAM. Types of CNC machines.

Interpolation for trajectory generation.

Integration in Flexible Fabrication Cells.

…

Chap. 6 – Discrete Event Systems [2 weeks]

Some pointers to CAD/CAM and CNC

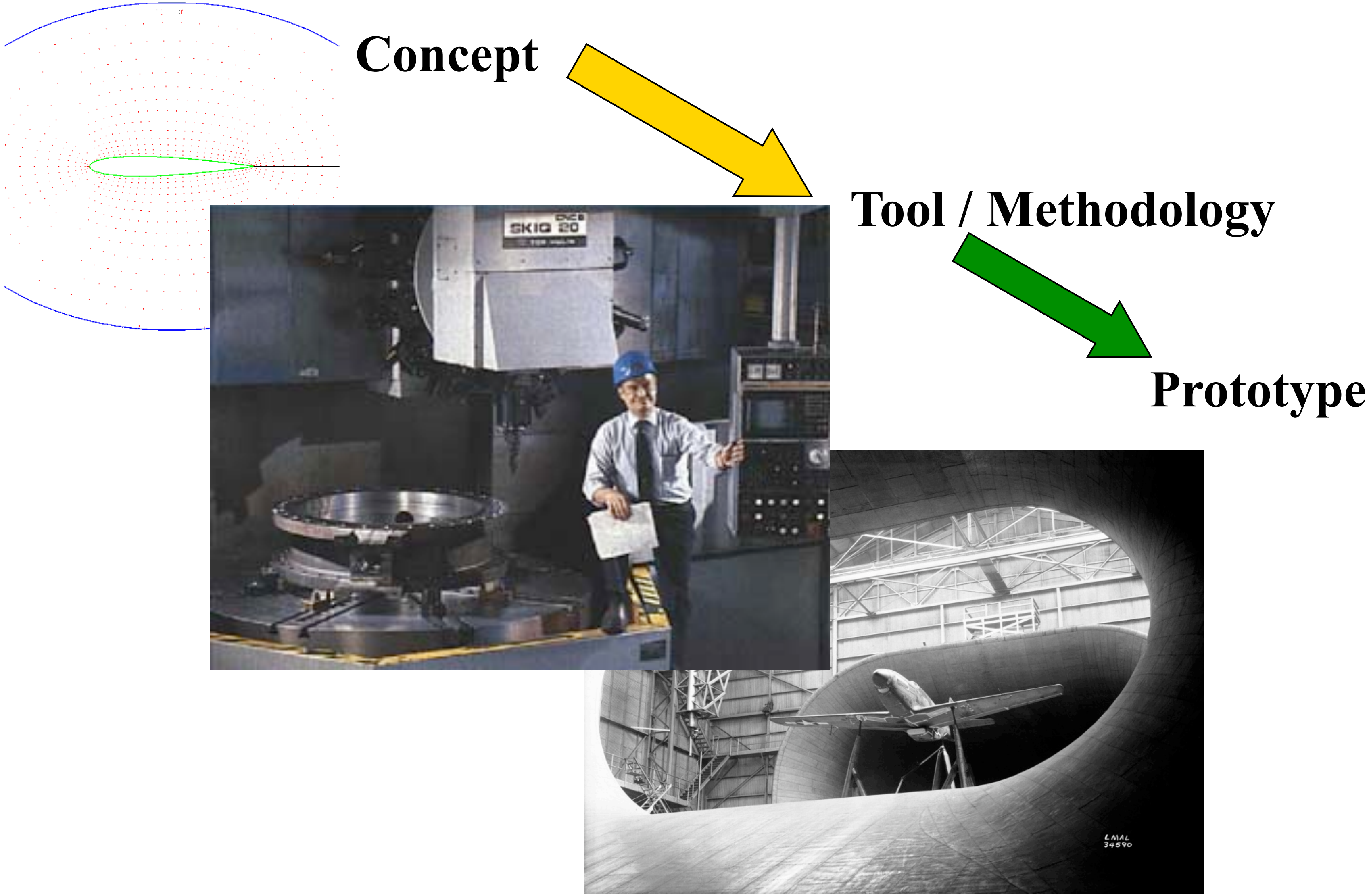
History: http://users.bergen.org/~jdefalco/CNC/history.html

Tutorial: http://users.bergen.org/~jdefalco/CNC/index.html
http://www-me.mit.edu/Lectures/MachineTools/outline.html
http://www.tarleton.edu/~gmollick/3503/lectures.htm

Editors (CAD): http://www.cncezpro.com/
http://www.cadstd.com/
http://www.turbocad.com
http://www.deskam.com/
http://www.cadopia.com/

Bibliography: * Computer Control of Manufacturing Systems, Yoram Koren, McGraw Hill, 1986.
* The CNC Workbook : An Introduction to Computer Numerical Control by Frank Nanfarra, et al.

CAD/CAM and CNC



Concept

Tool / Methodology

Prototype

Brief relevant history

NC

1947 – US Air Force needs lead John Parsons to develop a machine able to Produce parts describes in 3D.

1949 – Contract with *Parsons Corporation* to implement to proposed method.

1952 – Demonstration at MIT of a working machine tool(NC), able to produce parts resorting to simultaneous interpolation on several axes.

1955 – First NC machine tools reach the market.

1957 - NC starts to be accepted as a solution in industrial applications , with first machines starting to produce.

197x – Profiting from the microprocessor invention appears the CNC.

Evolution in brief

CAD/CAM and CNC

- Modification of existing machine tools with motion sensors and automatic advance systems.

- Close-loop control systems for axis control.

- Incorporation of the computational advances in the CNC machines.

- Development of high accuraccy interpolation algorithms to trajectory interpolation.

- Resort to CAD systems to design parts and to manage the use of CNC machines.

CAD/CAM e CNC

Objectives:

- To augment the accuraccy, reliability, and the ability to introduce changes/new designs.

- To augment the workload.

- To reduce prodution costs.

- To reduce waste due to errors and other human factors.

- To carry out complex tasks (e.g. Simlutaneous 3D interpolation).

- Augment precision of the produced parts.

CAD/CAM and CNC

Advantages:

- To reduce the production/delivery time.
- To reduce costs associated to parts and other auxiliary.
- To reduce storage space.
- To reduce time to start production.
- To reduce machining time.
- To reduce time to market (on the design/redesign and production).

CAD/CAM and CNC

Limitations:

- High initial investment (30.000 to 1.500.000 euros)
- Specialized maintenance required
- Does not eliminates the human errors completely.
- Requires more specialized operators.
- Not so relevant the advantages on the production of small or very small series.

CAD/CAM e CNC

Methodology CAD/CAM

To use technical data from a database in the design and production stages. Information on parts, materials, tools, and machines are integrated.

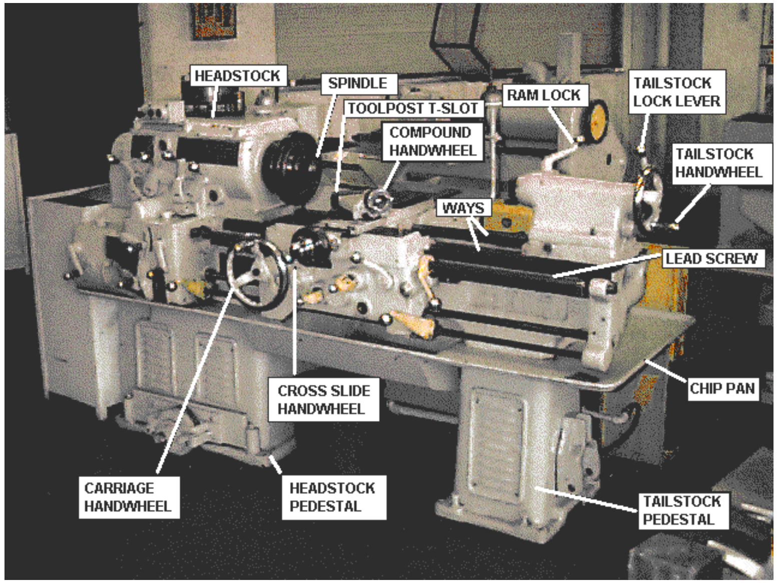
CAD (Computer Aided Design)

Allows the design in a computer environment.

CAM (Computer Aided Manufacturing)

To manage programs and production stages on a computer.

Bridgeport
TORQ-CUT
22

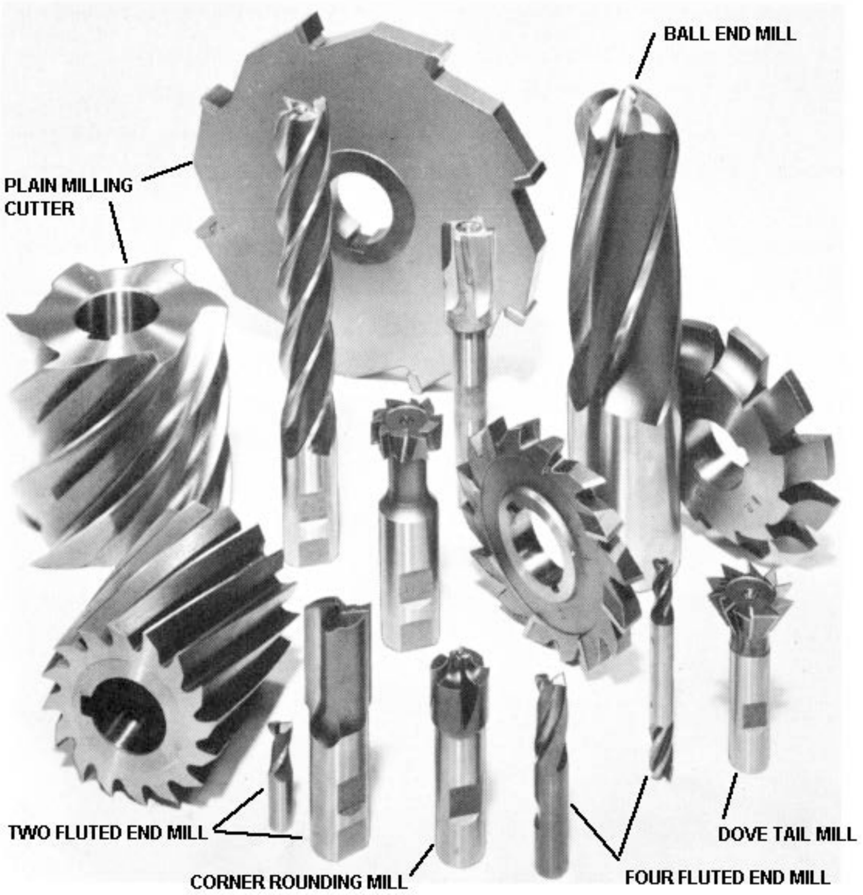

HEADSTOCK
SPINDLE
TOOLPOST T-SLOT
COMPOUND HANDWHEEL
RAM LOCK
TAILSTOCK LOCK LEVER
TAILSTOCK HANDWHEEL
WAYS
LEAD SCREW
CHIP PAN
CROSS SLIDE HANDWHEEL
CARRIAGE HANDWHEEL
HEADSTOCK PEDESTAL
TAILSTOCK PEDESTAL

SKIQ 20

X 0.000
Y 0.000
Z 0.000

CAD/CAM and CNC

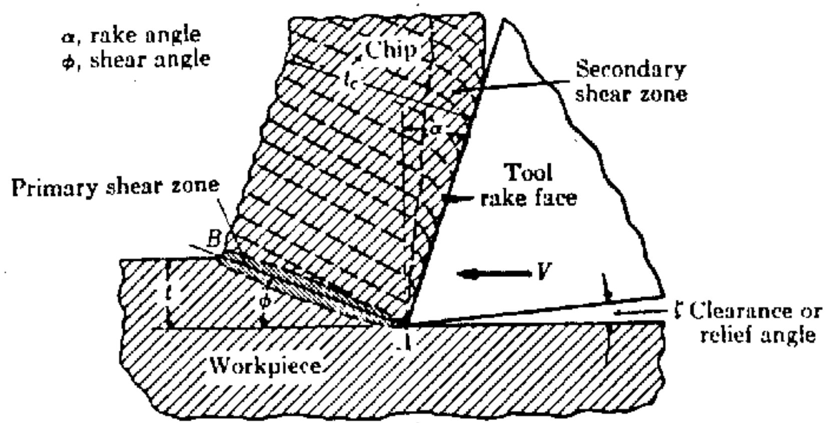
Tools:

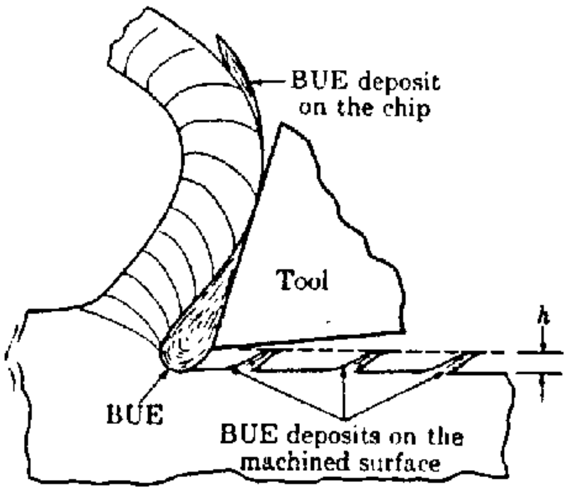


CAD/CAM and CNC

Tools:

Atention to the constraints on the materials used!...

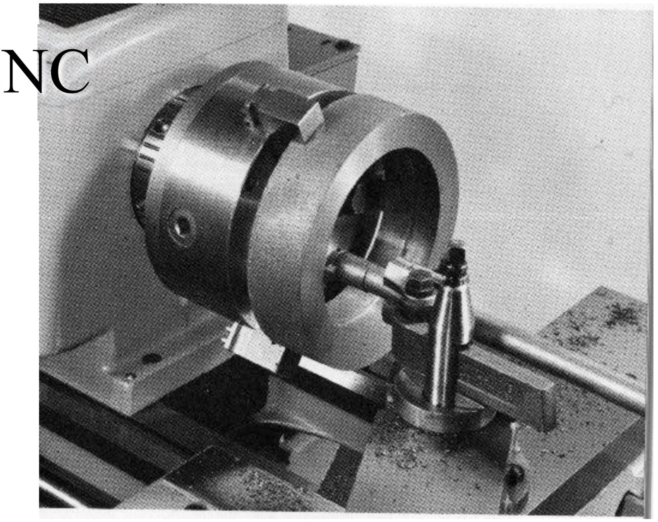


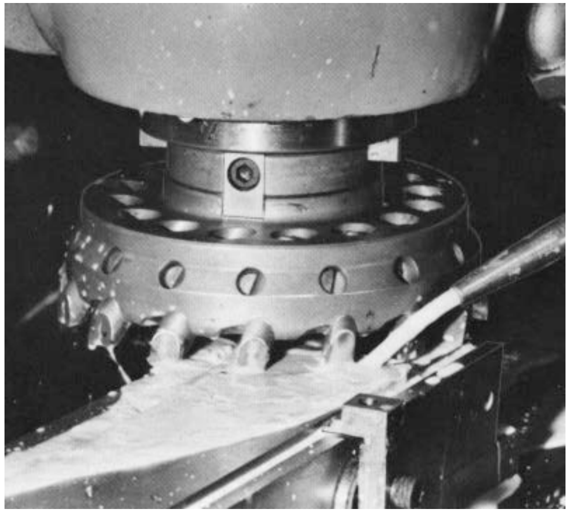


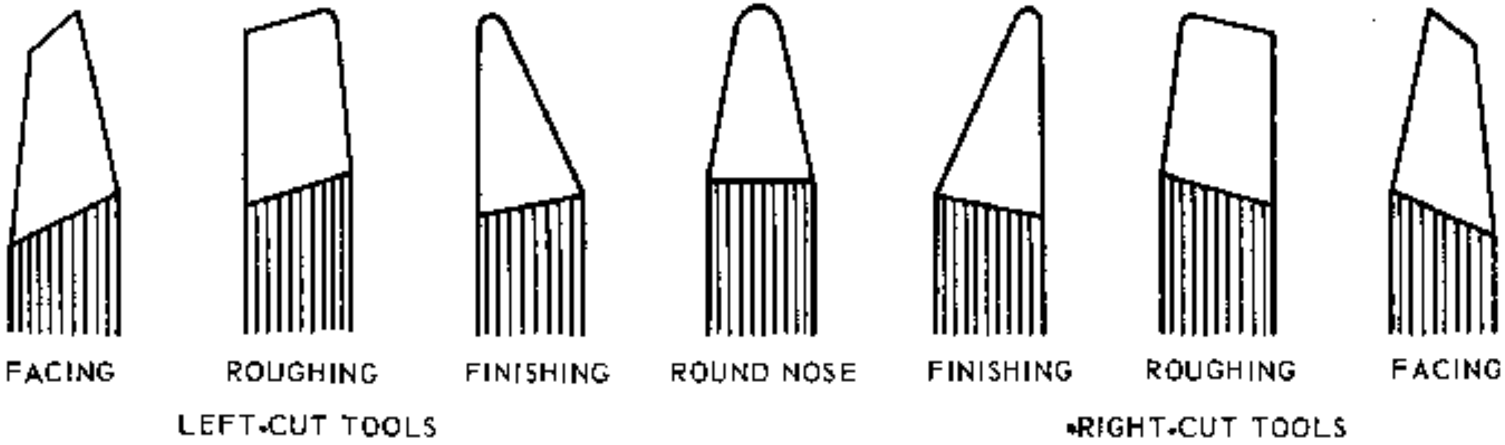
- **Speed of advance**
- **Speed of rotation**
- **Type of tool**

CAD/CAM and CNC

Tools:







Specific tools to perform different operations.

CAD/CAM and CNC

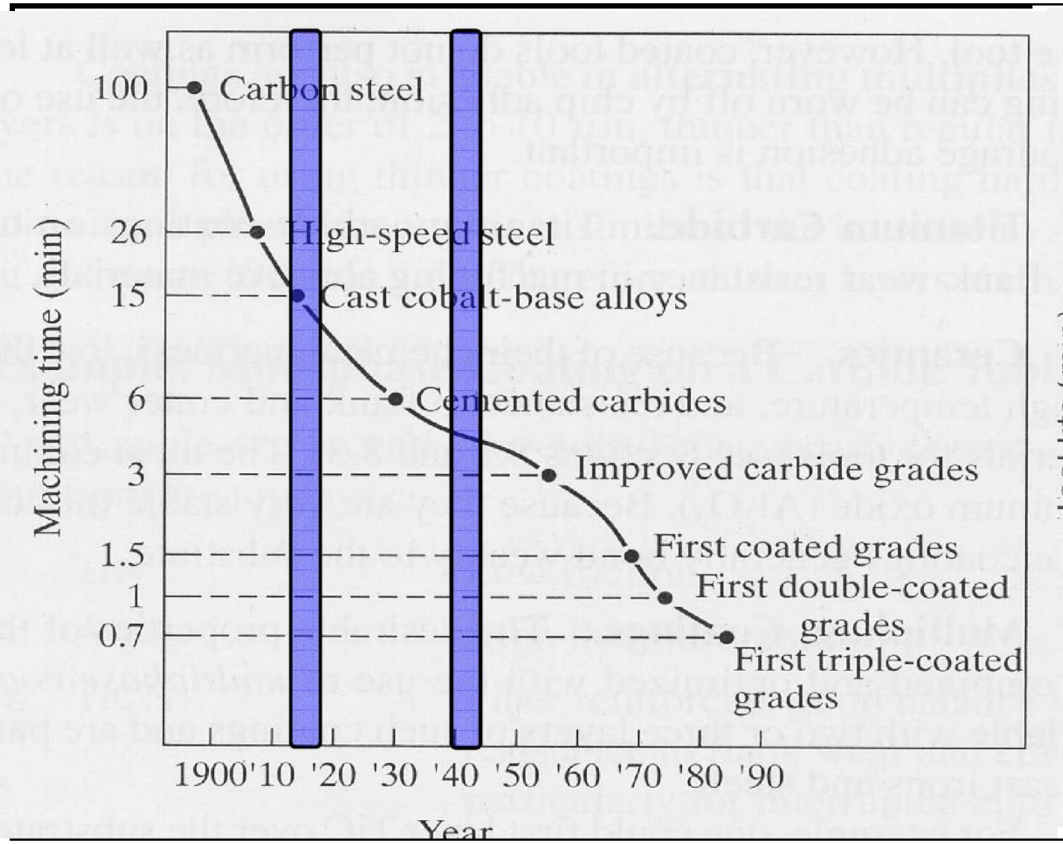
Tools: impact on the quality of finishing (mm)

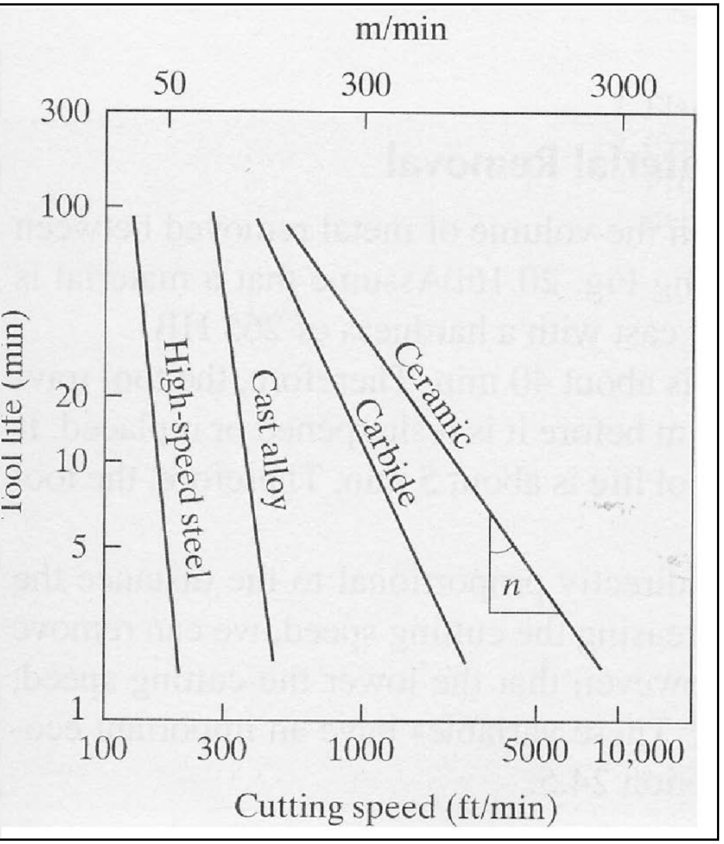
ama

Método	50	25	12	6	3	1.5	.8	.4	.2	.1	.05	.025	.0125
Flame cut	▨	■	▨										
Sawing		▨	■	■	▨								
Planeing		▨	■	■	■	▨							
Drilling			▨	■	■	▨							
Chemical machinning			▨	■	■	▨							
Electrical dischage				▨	■	▨							
Milling		▨	▨	■	■	■	▨	▨					
Augment drilling				▨	■	■	▨						
Electron beam				■	■	■	▨	▨					
LASER cut			▨	▨	■	■	■	■	▨	▨			
Electrochemical cut			▨	■	■	■	■	▨					
Lath						▨	■	■	▨	▨			
Electrolitical machining							▨	■	▨	▨			
Exctrusion					▨	▨	■	■	■	▨	▨		
“Afiar”						▨	■	■	■	▨	▨		
Polishing							▨	■	■	▨	▨	▨	
“Quinar”							▨	■	■	■	▨	▨	

CAD/CAM and CNC

Evolution of tools performance:





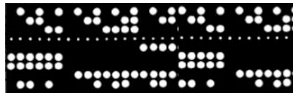
CAD/CAM and CNC

Industrial areas of application:

- **Aerospace**
- **Maquinery**
- **Electricity (board production)**
- **Automobiles**
- **Instrumentation**
- **Moulds**

CAD/CAM and CNC

Evolution of Numerical Control



- Numerical Control (NC)
  - Data on paper ou received in serial port
  - NC machine unable to perform computations
  - Hardware interpolation

- Direct Numerical Control (DNC)
  - Central computer control a number of machines DNC ou CNC

- Computer Numerical control (CNC)
  - A computer is on the core of each machine tool
  - Computation and interoplation algorithms run on the machine

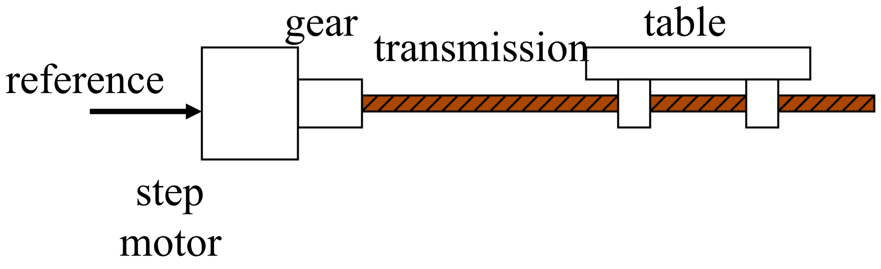
- Distributive numerical control
  - scheduling
  - Quality control
  - Remore monitoring

CAD/CAM and CNC

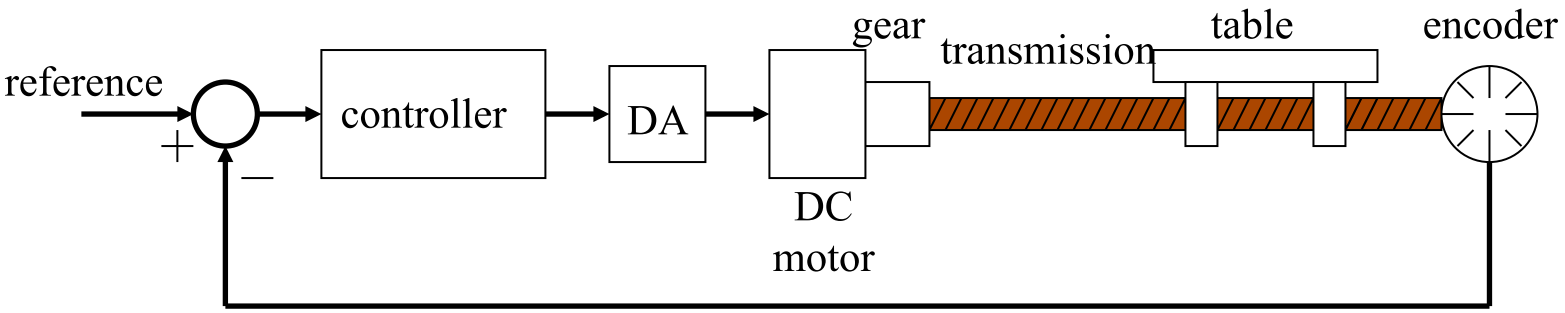
Numeric Control

Architecture of a NC system

Open-loop

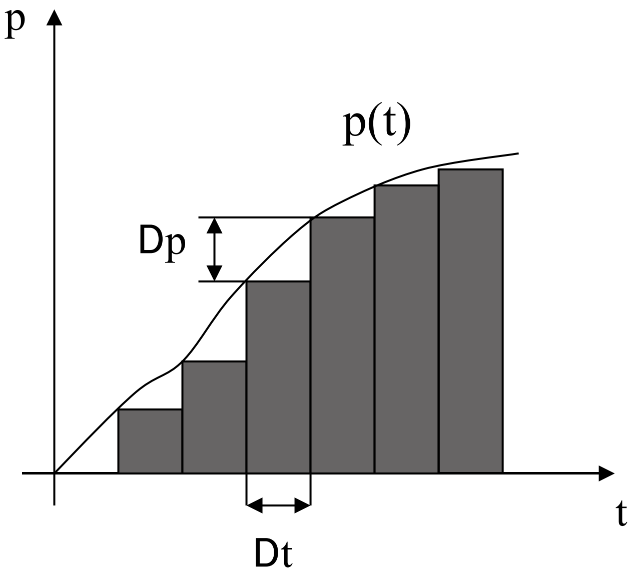


Close-loop



CAD/CAM and CNC

Interpolation
Motivation: numerical integration



Area of a function

$$z(t)=\int_0^t p(\tau)d\tau \cong \sum_{i=1}^{k} p_i\Delta t$$

Introducing z_k, as the value of z at t=kDt

$$z_k=\sum_{i=1}^{k-1} p_i\Delta t + p_k\Delta t = z_{k-1}+\Delta z_k , \qquad \Delta z_k = p_k\Delta t$$

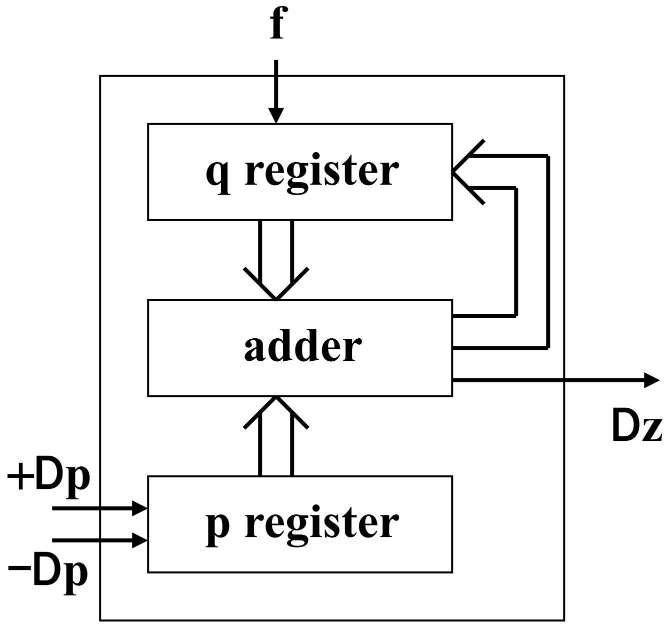
The integrator works at a rythm of f=1/Dt and the function p is given app. by:

$$p_k = p_{k-1} \pm \Delta p_k$$

To be able to implement the integrator in registers with n bits, p must verify $p_k<2^n$.

CAD/CAM and CNC

Implementation of a DDA
Digital Differential Analyzer



The p register input is +1, 0 ou –1.

The q register stores the area integration value

$$q_k = q_{k-1} + p_k.$$

If the q register value exceeds (2^n-1), and overflow occurs and Dz=1:

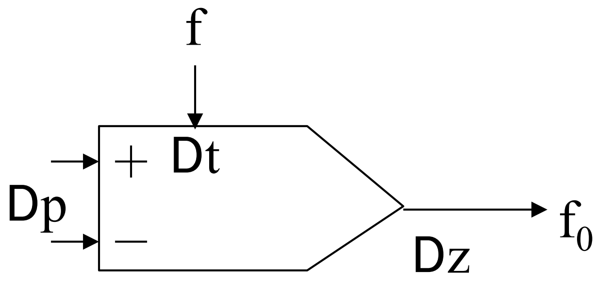
$$\Delta z_k = 2^{-n} p_k$$

Defining C=f/2^n, and given that f=1/Dt:

$$\Delta z_k = C p_k \Delta t$$

CAD/CAM and CNC

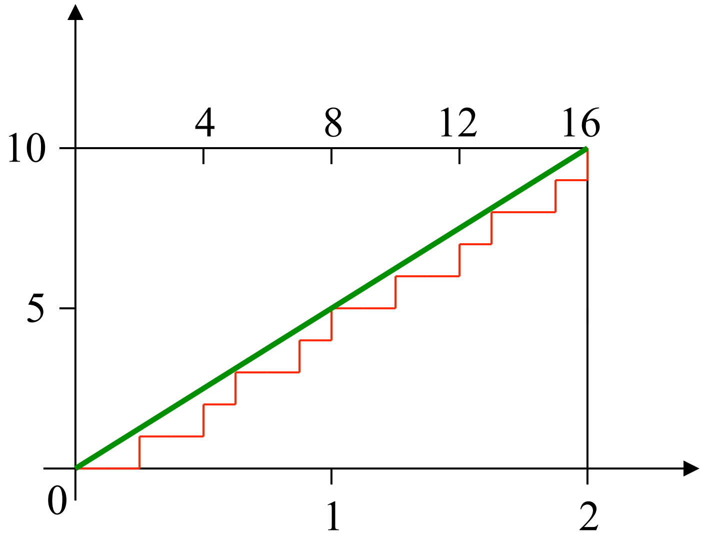
DDA for Linear Interpolation:



Let q=5 and assume 3 bits registers

Passo	q	Dz	S Dz
1	5		0
2	2	1	1
3	7		1
4	4	1	2
5	1	1	3
6	6		3
7	3	1	4
8	0	1	5
9	5		5
		...	

$$f_0 = \left(\frac{\Delta z}{\Delta t}\right)_k = C p_k, \qquad \text{where} \qquad C = \frac{f}{2^n}$$



CAD/CAM and CNC

Exponential Deacceleration:

Let $p(t)=p_0e^{-\alpha t}$ and $\dfrac{\Delta z}{\Delta t}=Cp_k=Cp_0e^{-\alpha t}$.

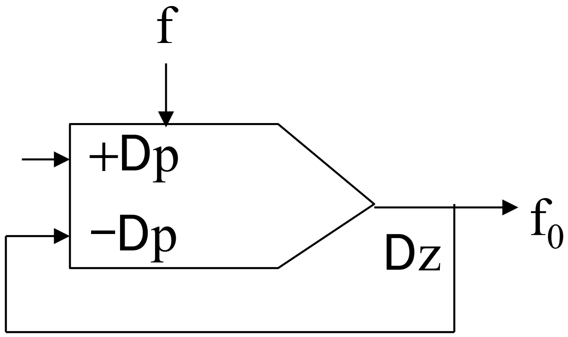
The differential of p(t) is appr.

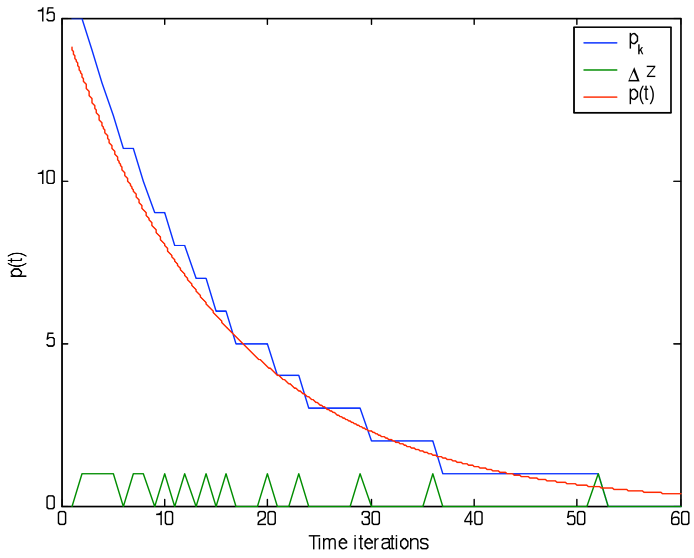
$$-\Delta p=\alpha p_k\Delta t$$

Example: $p(t)=15e^{-t}$

Setting C=a,

$$-\Delta p=\Delta z$$





CAD/CAM and CNC

Circular Interpolation:

Let $(X-R)^2+Y^2=R^2$ or

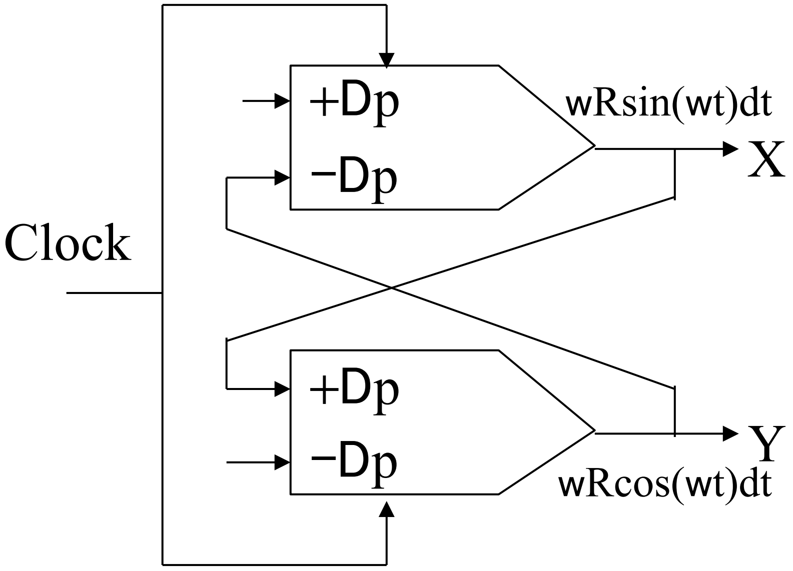
$$X=R(1-\cos(\omega t))$$

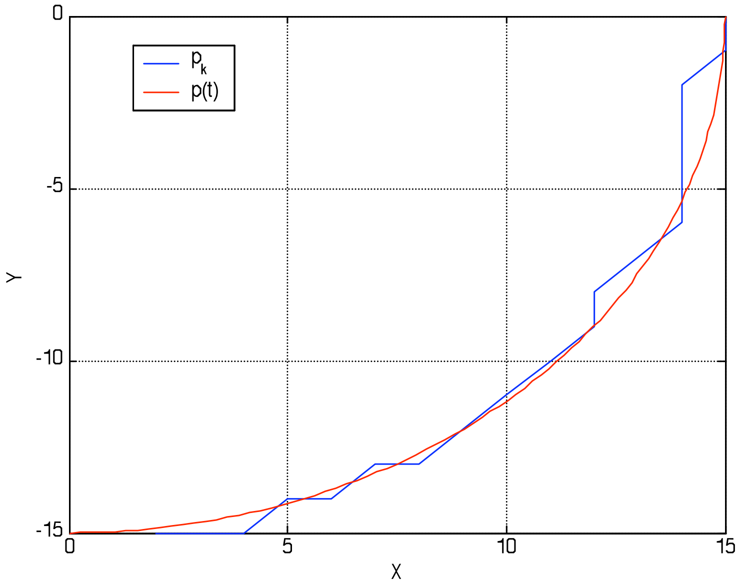
$$Y=\quad R\sin(\omega t)$$

The differential is

$$dX=\omega R\sin(\omega t)dt=d(-R\cos(\omega t))$$

$$dY=\omega R\cos(\omega t)dt=\ d(R\sin(\omega t))$$

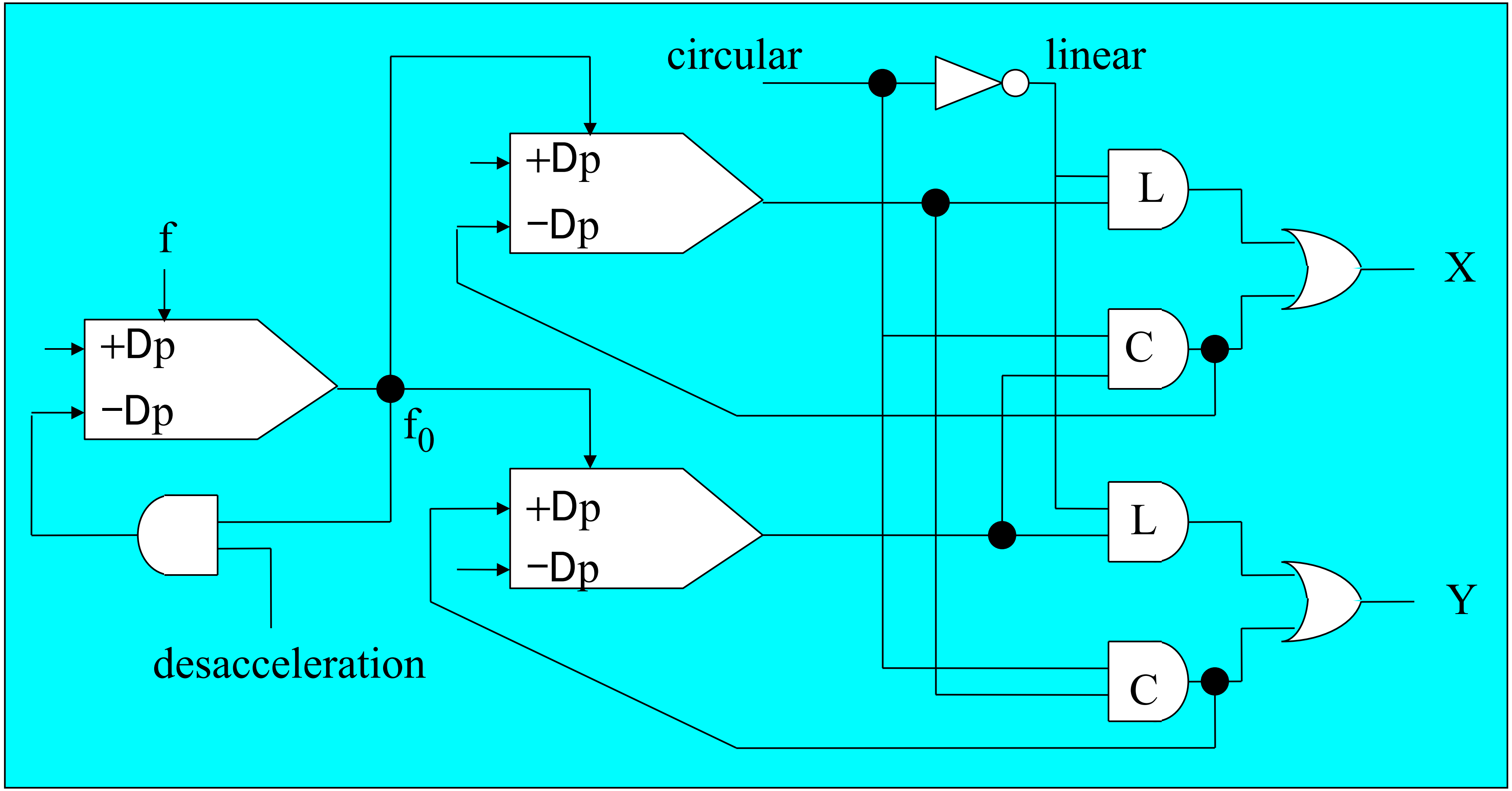




Example: Circunference of radius 15, centered at the origin.

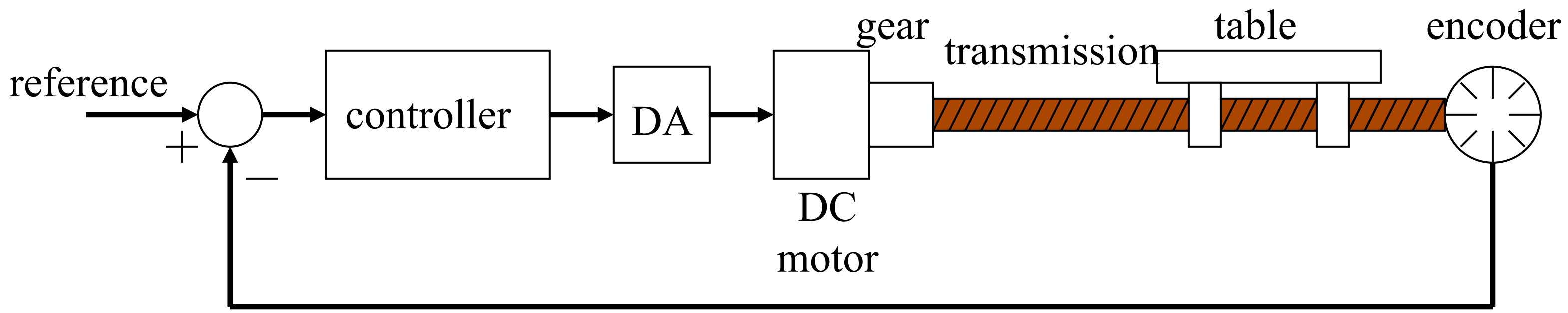
CAD/CAM and CNC

Full DDA

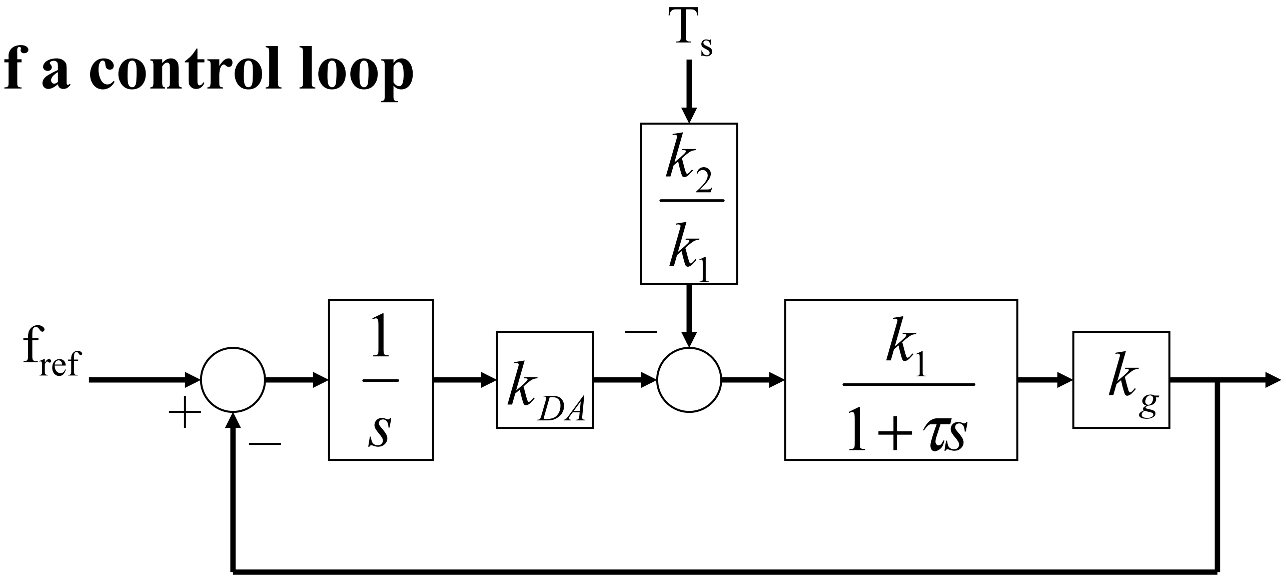


CAD/CAM and CNC

CNC Axes Control



Dynamics of a control loop

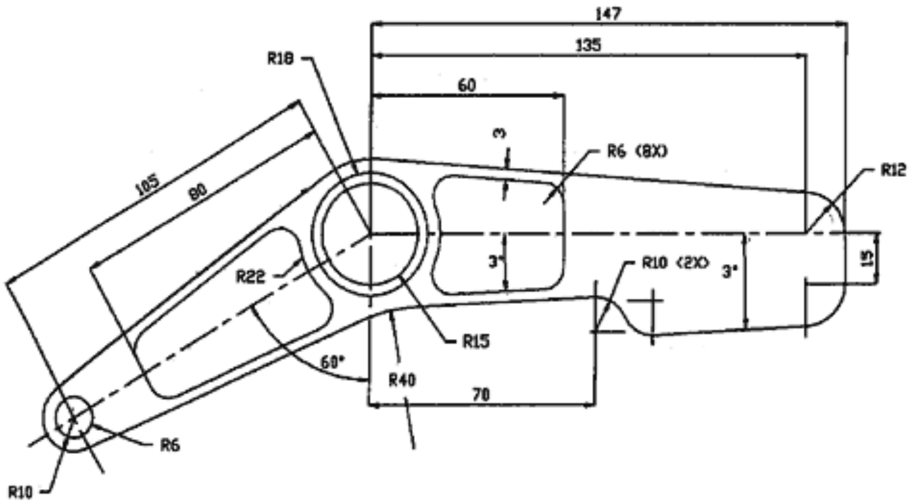


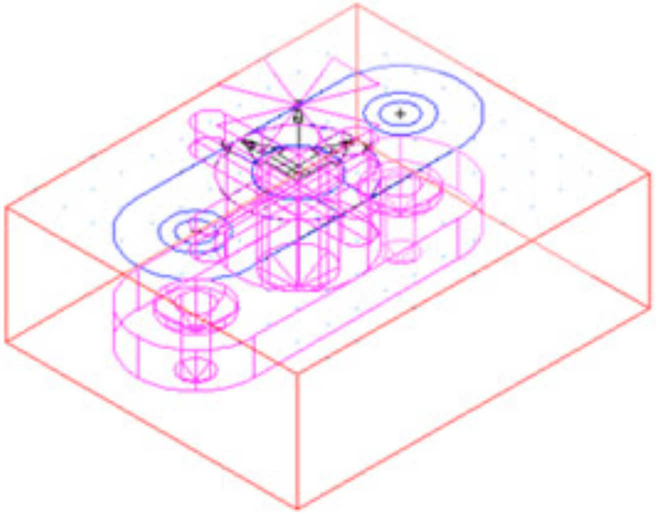
CAD/CAM and CNC

CNC Programming

Steps to execute a part

A) Read/interpret the technical drawings





CAD/CAM and CNC

CNC Programming

B) Choice of the most adequate machine tool for the several stages of machinning

Relevant features:

- **The workspace of a machine versus the part to be produced**
- **The options available on each machine**
- **The tools available**
- **The mounting and the part handling**
- **The operations that each machine can perform**

CAD/CAM and CNC

CNC Programming

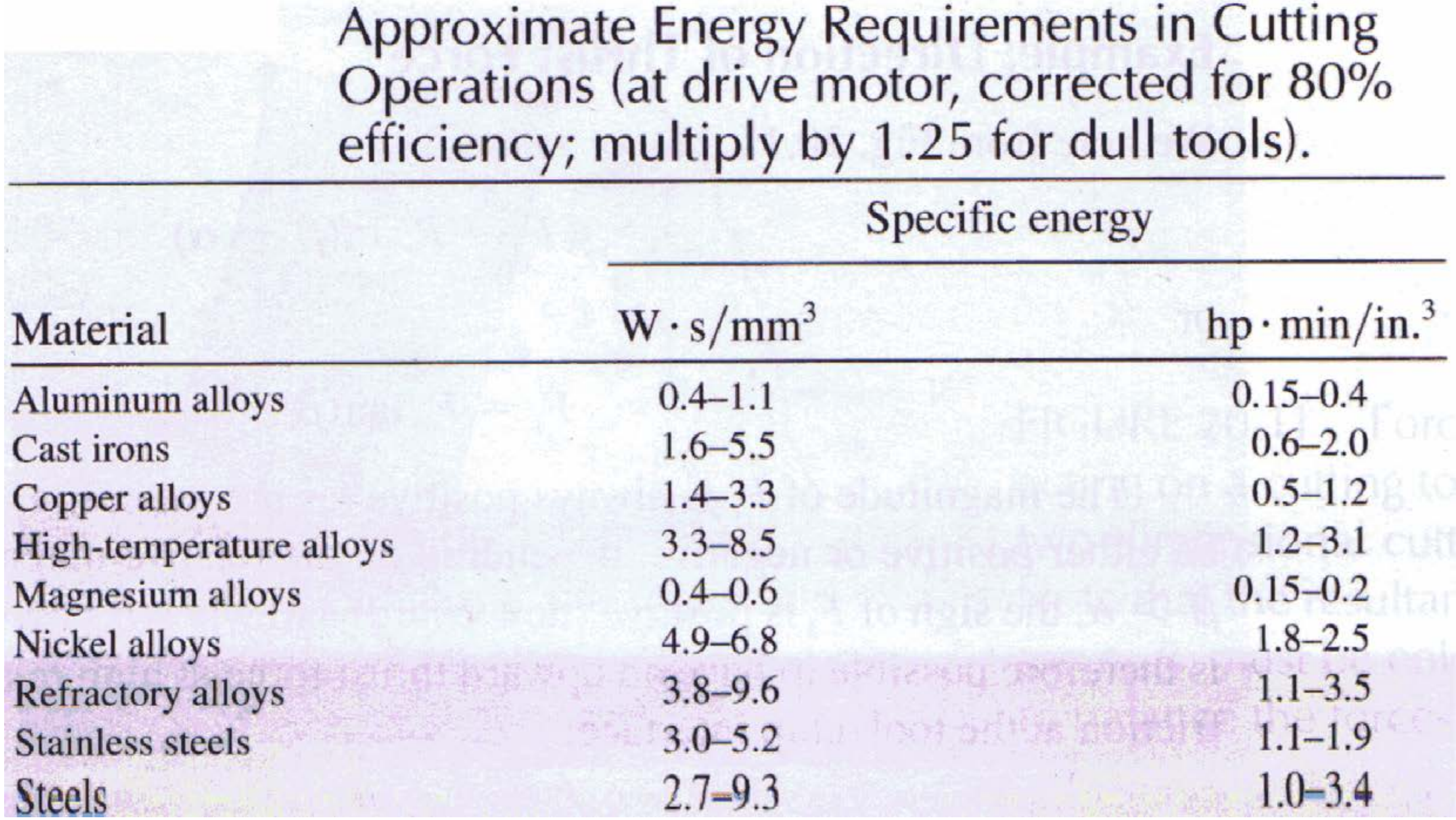
C) Choice of the most adequate tools

Relevant features:

- **The material to be machinned and its characteristics**
- **Standard tools cost less**
- **The quality of the mounting part is function of the number od parts to produce**
- **Use the right tool for the job**
- **Verify if there are backup tools and/or stored available**
- **Take into account tool aging**

CAD/CAM and CNC

CNC Programming



Approximate Energy Requirements in Cutting Operations (at drive motor, corrected for 80% efficiency; multiply by 1.25 for dull tools).

	Specific energy	
Material	$W \cdot s/mm^3$	$hp \cdot min/in.^3$
Aluminum alloys	0.4–1.1	0.15–0.4
Cast irons	1.6–5.5	0.6–2.0
Copper alloys	1.4–3.3	0.5–1.2
High-temperature alloys	3.3–8.5	1.2–3.1
Magnesium alloys	0.4–0.6	0.15–0.2
Nickel alloys	4.9–6.8	1.8–2.5
Refractory alloys	3.8–9.6	1.1–3.5
Stainless steels	3.0–5.2	1.1–1.9
Steels	2.7–9.3	1.0–3.4

CAD/CAM and CNC

CNC Programming

D) Cutting data

- **Spindle Speed – speed of rotation of the cutting tool (rpm)**

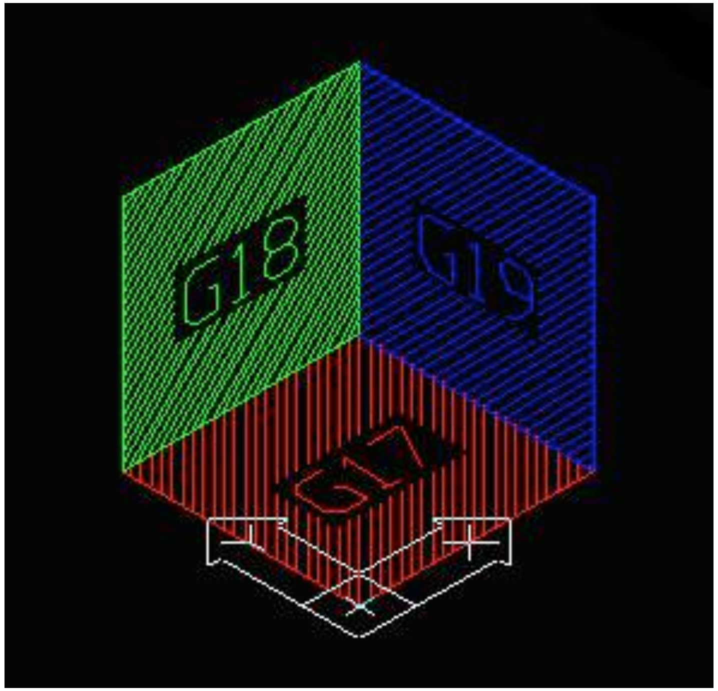
- **Feedrate – linear velocity of advance to machine the part (mm/ minute)**

- **Depth of Cut –deth of machinning in z (mm)**

CAD/CAM and CNC

CNC Programming

E) Choice of the interpolation plane, in 2D ½ machines



CAD/CAM and CNC

CNC Programming

F_1) Unit system

imperial –inches (G70) or international milimeters (G71).

F_2) Command mode*

Absolut – relative to world coordinate system (G90)

Relative– mouvement relative to the actual position (G91)

*** There are other command modes, e.g. helicoidal.**

CAD/CAM and CNC

CNC Programming

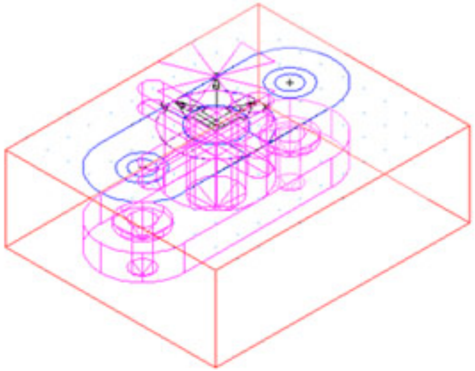
G) MANUAL DATA INPUT

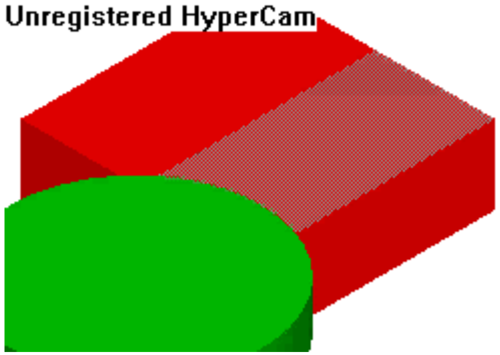
N	Sequence Number
G	Preparatory Functions
X	X Axis Command
Y	Y Axis Command
Z	Z Axis Command
R	Radius from specified center
A	Angle ccw from +X vector
I	X axis arc center offset
J	Y axis arc center offset
K	Z axis arc center offset
F	Feedrate
S	Spindle speed
T	Tool number
M	Miscellaneous function

CAD/CAM and CNC

Example of a CNC program

```
N30 G0 T1 M6
N35 S2037 M3
N40 G0 G2 X6.32 Y-0.9267 M8
N45 Z1.1
N50 Z0.12
N55 G1 Z0. F91.7
N60 X-2.82
N65 Y0.9467
N70 X6.32
N75 Y2.82
N80 X-2.82
N85 G0 Z1.1
...
```

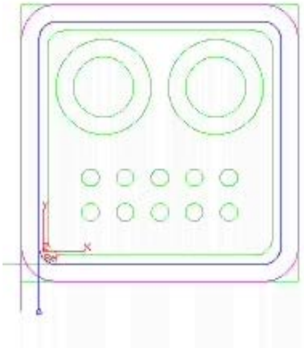




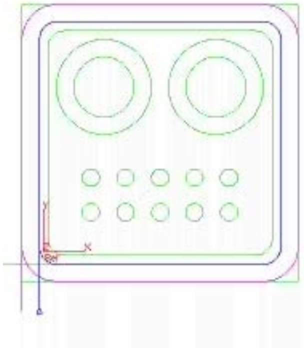
CAD/CAM and CNC

Preparatory functions (inc.)

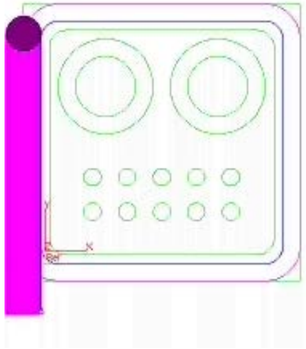
G00 – GO



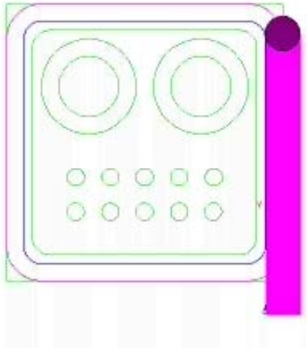
G01 – Linear Interpolation



G02 – Circular Interpolation (CW)



G03 – Circular Interpolation (CCW)



CAD/CAM and CNC

Other preparatory functions

- G04 - A temporary dwell, or delay in tool motion.
- G05 - A permanent hold, or stopping of tool motion. It is canceled by the machine operator.
- G22 - Activation of the stored axis travel limits, which are used to establish a safety boundary.
- G23 - Deactivation of the stored axis travel limits.
- G27 - Return to the machine home position via a programmed intermediate point
- G34 - Thread cutting with an increasing lead.
- G35 - Thread cutting with a decreasing lead.
- G40 - Cancellation of any previously programmed tool radius compensation
- G42 - Application of cutter radius compensation to the right of the workpiece with respect to the direction of tool travel.
- G43 - Activation of tool length compensation in the same direction of the offset value
- G71 - Canned cycle for multiple-pass turning on a lathe (foreign-made)
- …

CAD/CAM and CNC

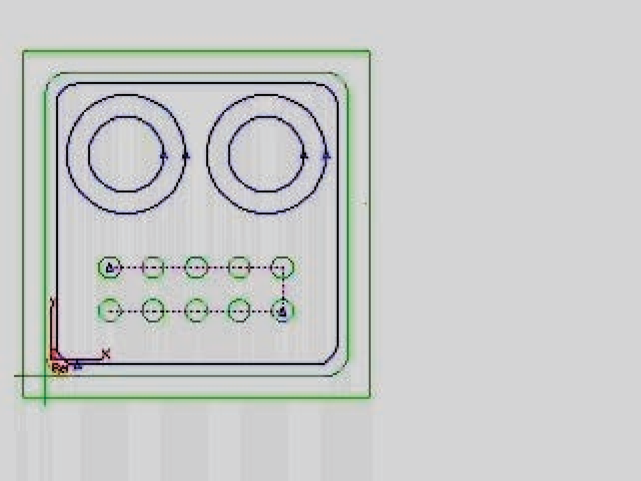
Miscelaneous functions

- M02 - Program end
- M03 - Start of spindle rotation clockwise
- M04 - Start of spindle rotation counterclockwise
- M07 - Start of mist coolant
- M08 - Start of flood coolant

CAD/CAM and CNC

Canned Cycles

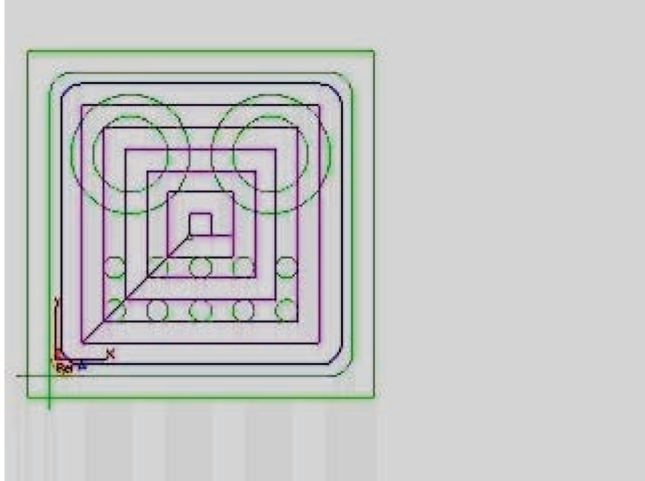
G81 – Drilling cycle with multiple holes



CAD/CAM and CNC

Ciclos Especiais or Canned Cycles

G78 – Rectangular pocket cycle, used to clean a square shaped area



CAD/CAM and CNC

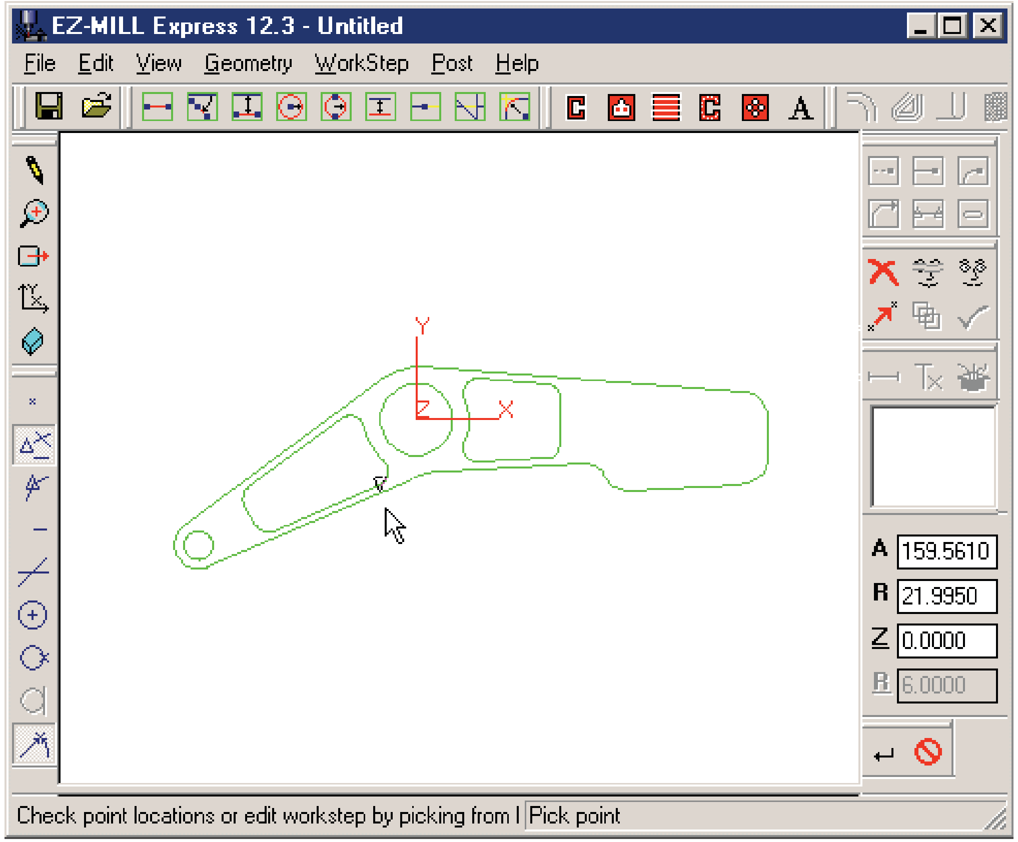
Tool change

Note: should be of easy access, when performed manually.

CAD/CAM and CNC

Example of CNC programming

Ver: http://www.ezcam.com/web/tour/tour.htm



EZ-MILL Express 12.3 - Untitled

A 159.5610

R 21.9950

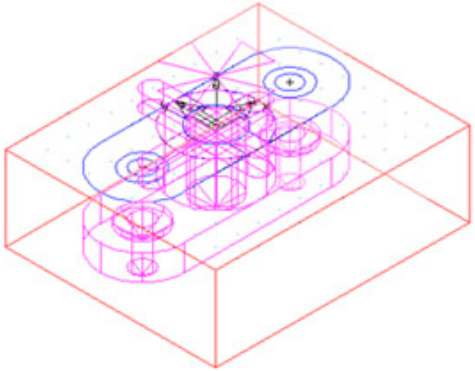
Z 0.0000

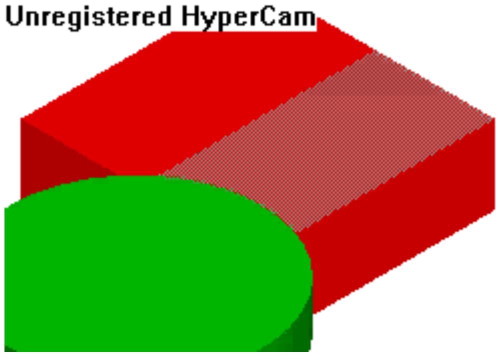
R 6.0000

Check point locations or edit workstep by picking from l | Pick point

CAD/CAM and CNC

Example of CNC programming





CAD/CAM and CNC

Advanced CNC programming languages

- Automatically program tool (APT)
  Desveloped at MIT in 1954

- Derived from APT:
  ADAPT (IBM)
  IFAPT (France)
  MINIAPT (Germany)

- Compact II

- Autospot

- SPLIT

CAD/CAM and CNC

Machine operation

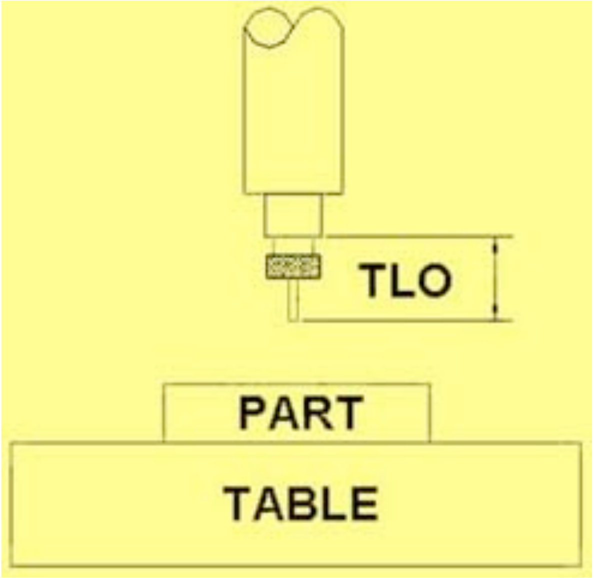
Rules of Security

- Security is essential!
- The eyes must be always protected.
- The tools and parts must be handled and installed properly.
- Avoid the use of large cloths
- Cleand the parts with a brush. Never with the hands.
- Be careful with you and the others.

CAD/CAM and CNC

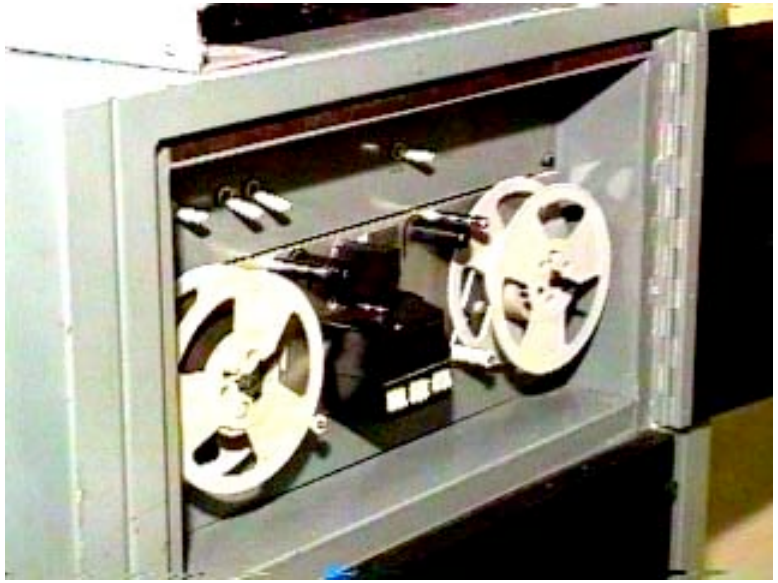
Machine operation

Verify tolerances and tools offsets for proper operation



CAD/CAM and CNC

Machine operation



Load program

Follow up machine operation

Verify carefully the produced part.

Industrial Automation

(Automação de Processos Industriais)

Discrete Event Systems

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

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

Syllabus:

Chap. 5 – CAD/CAM and CNC [1 week]

…

Chap. 6 – Discrete Event Systems [2 weeks]

Discrete event systems modeling. Automata.

Petri Nets: state, dynamics, and modeling.

Extended and strict models. Subclasses of Petri nets.

…

Chap. 7 – Analysis of Discrete Event Systems [2 weeks]

Some pointers to Discrete Event Systems

History: http://prosys.changwon.ac.kr/docs/petrinet/1.htm

Tutorial: http://vita.bu.edu/cgc/MIDEDS/
http://www.daimi.au.dk/PetriNets/

Analyzers, and Simulators: http://www.ppgia.pucpr.br/~maziero/petri/arp.html (in Portuguese)
http://wiki.daimi.au.dk:8000/cpntools/cpntools.wiki
http://www.informatik.hu-berlin.de/top/pnk/download.html

Bibliography:

* Cassandras, Christos G., "Discrete Event Systems - Modeling and Performance Analysis", Aksen Associates, 1993.
* Peterson, James L., "Petri Net Theory and the Modeling of Systems", Prentice-Hall,1981.
* Petri Nets and GRAFCET: Tools for Modelling Discrete Event Systems R. DAVID, H. ALLA, New York : PRENTICE HALL Editions, 1992

Generic characterization of systems resorting to input / output relations

State equations:

$$\dot{x}(t) = f(x(t), u(t), t)$$
$$y(t) = g(x(t), u(t), t)$$

in continuous time (or in discrete time)

Examples?

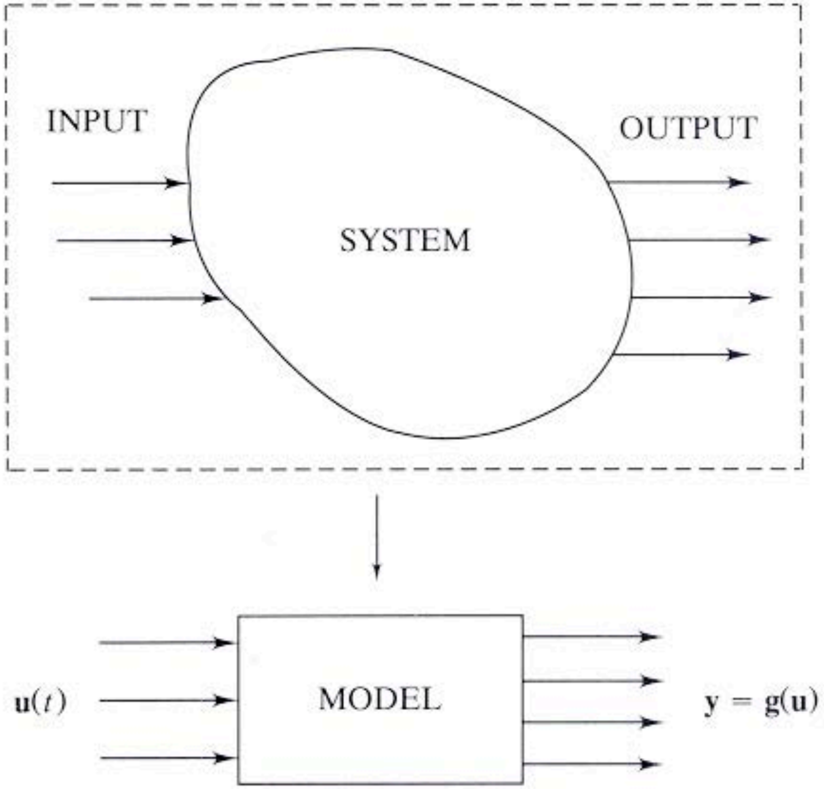


Figure 1.1. Simple modeling process.

Open loop vs close-loop (⇔ the use of feedback)

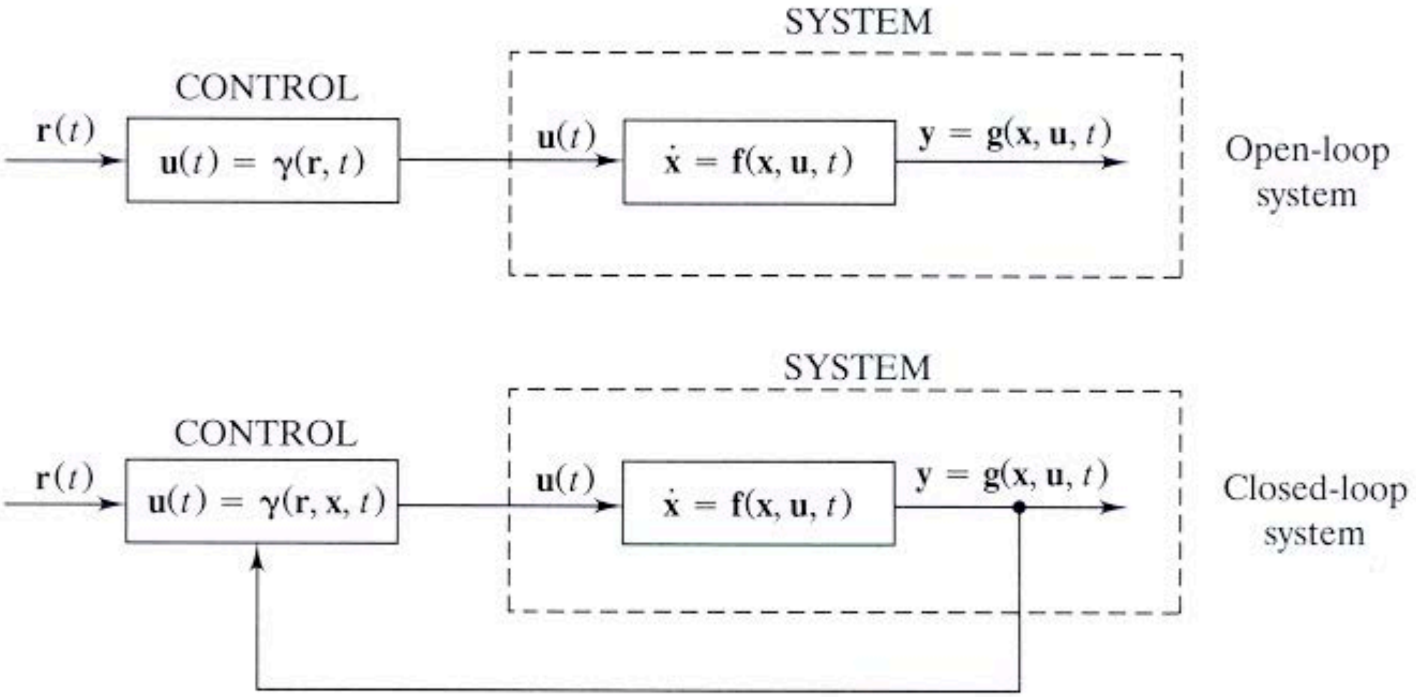


Figure 1.17. Open-loop and closed-loop systems.

Advantages of feedback?

(to revisit during SEDs supervision study)

Example of close-loop with feedback

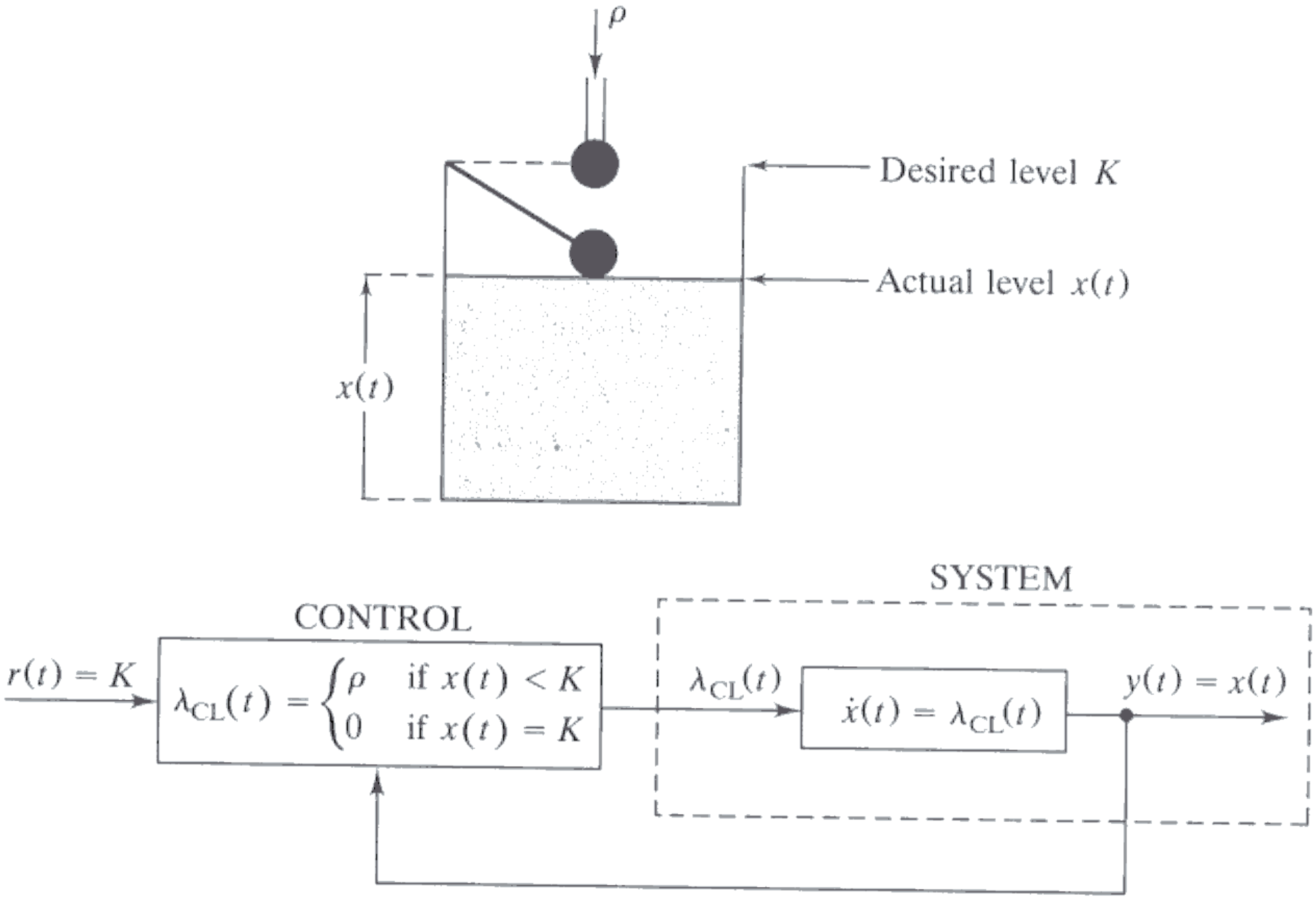

ρ
Desired level K
Actual level $x(t)$
$x(t)$
SYSTEM
CONTROL
$r(t) = K$
$\lambda_{CL}(t) = \begin{cases} \rho & \text{if } x(t) < K \\ 0 & \text{if } x(t) = K \end{cases}$
$\lambda_{CL}(t)$
$\dot{x}(t) = \lambda_{CL}(t)$
$y(t) = x(t)$

Figure 1.18. Flow system of Example 1.11 and closed-loop control model.

Discrete Event Systems: Examples

Set of events:

E={N, S, E, W}

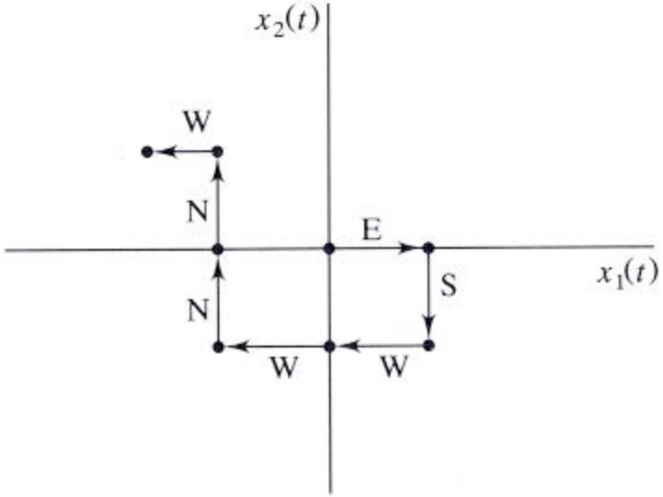


Figure 1.20. Random walk on a plane for Example 1.12.

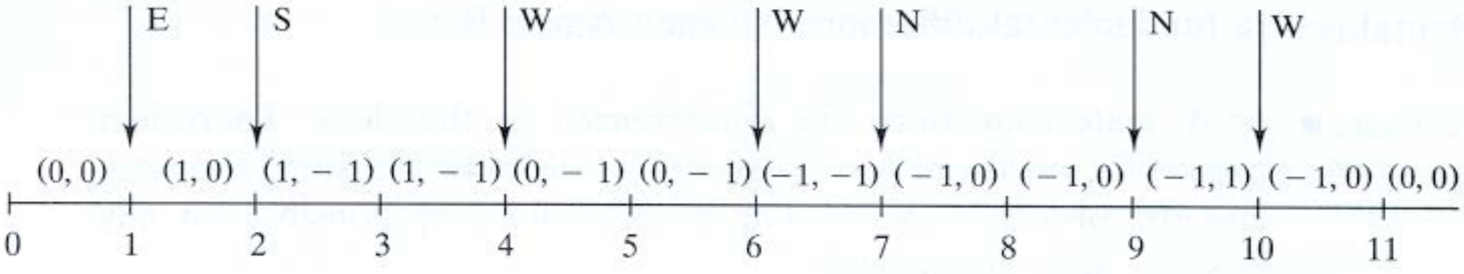


Figure 1.21. Event-driven random walk on a plane.

Characteristics of systems with continuous variables

1.State space is continuous

2. The state transition mechanism is *time-driven*

Characteristics of systems with discrete events

1.State space is discrete

2. The state transition mechanism is *event-driven*

Polling is avoided!

Taxonomy of Systems

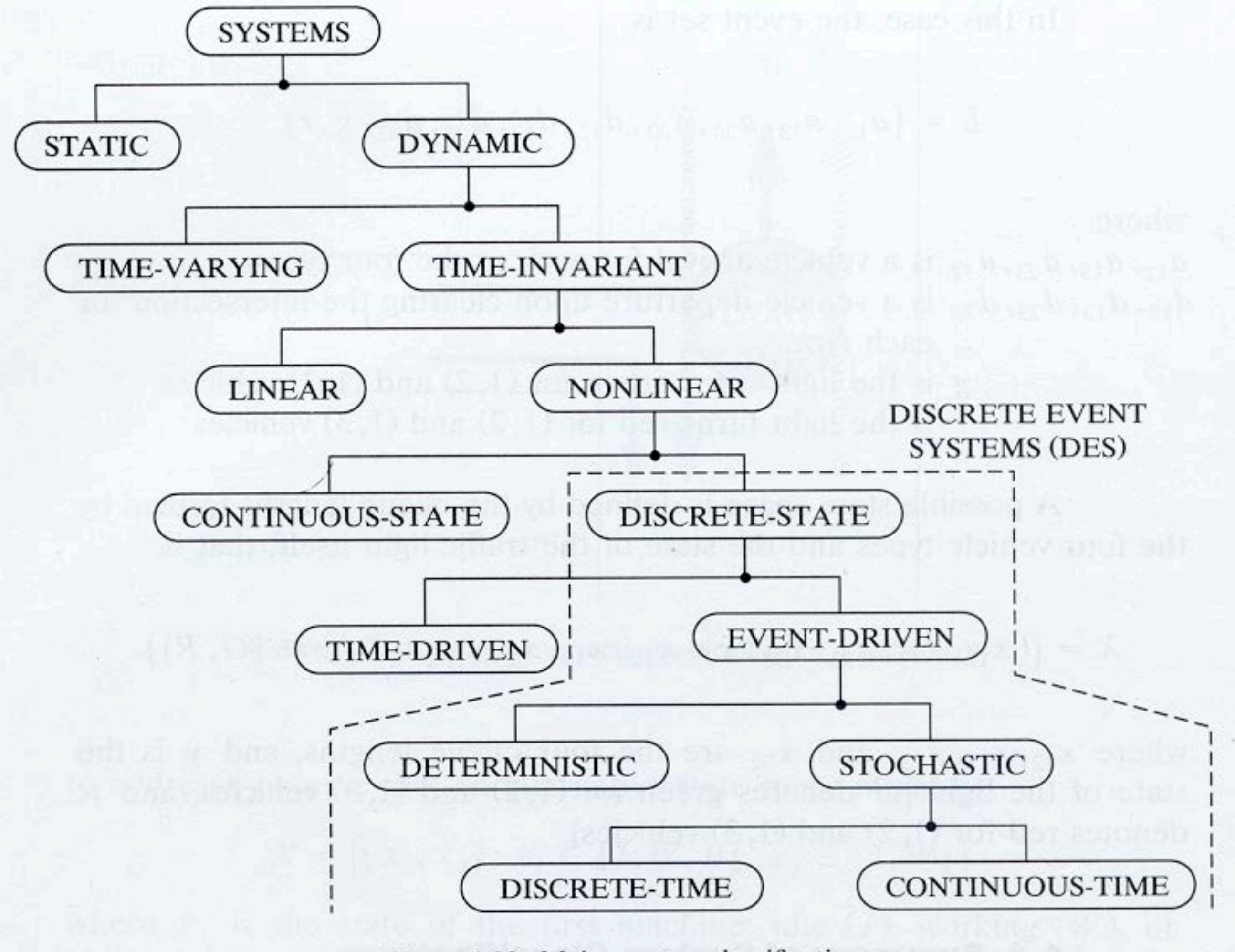
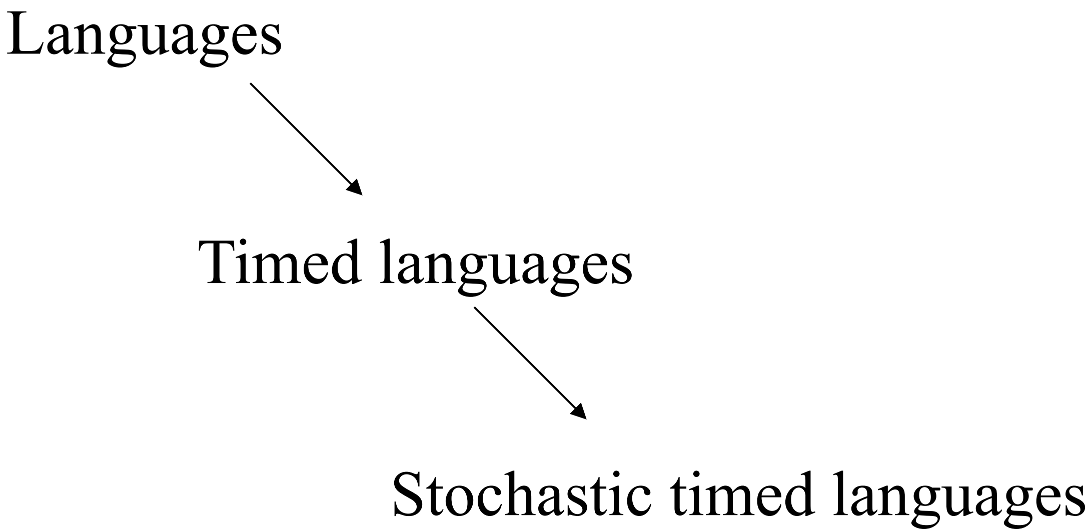


Figure 1.29. Major system classifications.

Levels of abstraction in the study of Discrete Event Systems



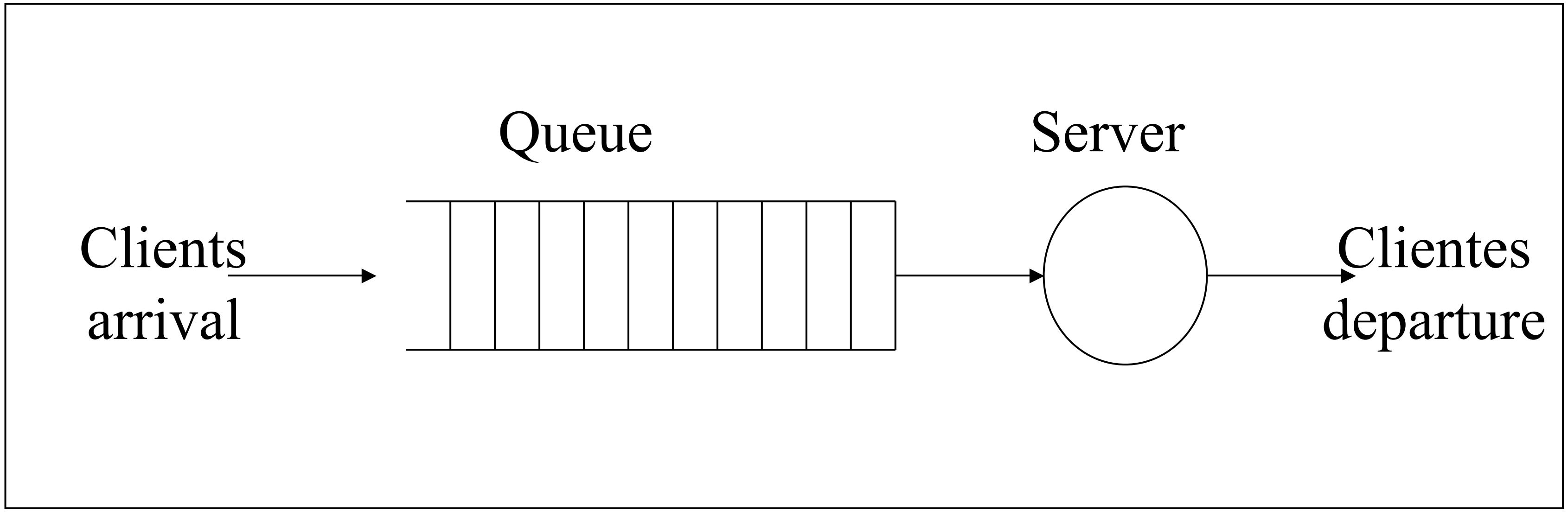
Languages

Timed languages

Stochastic timed languages

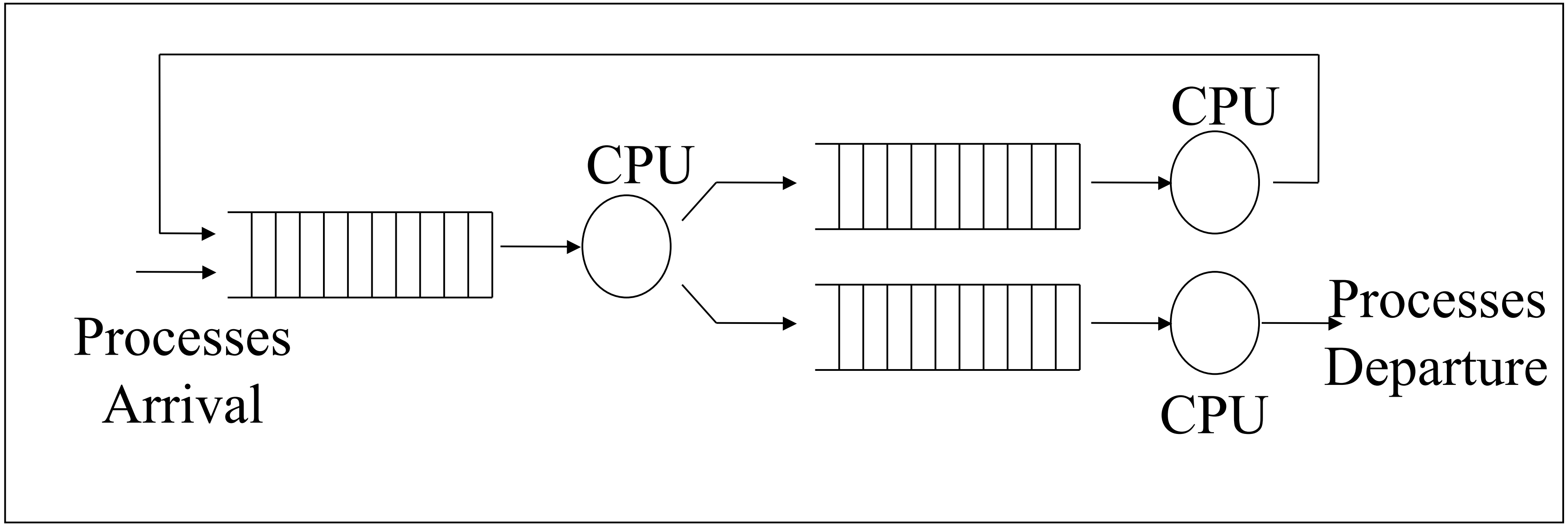
Discrete Event Systems: Examples

Queueing systems



Discrete Event Systems: Examples

Computational Systems



Systems' Theory Objectives

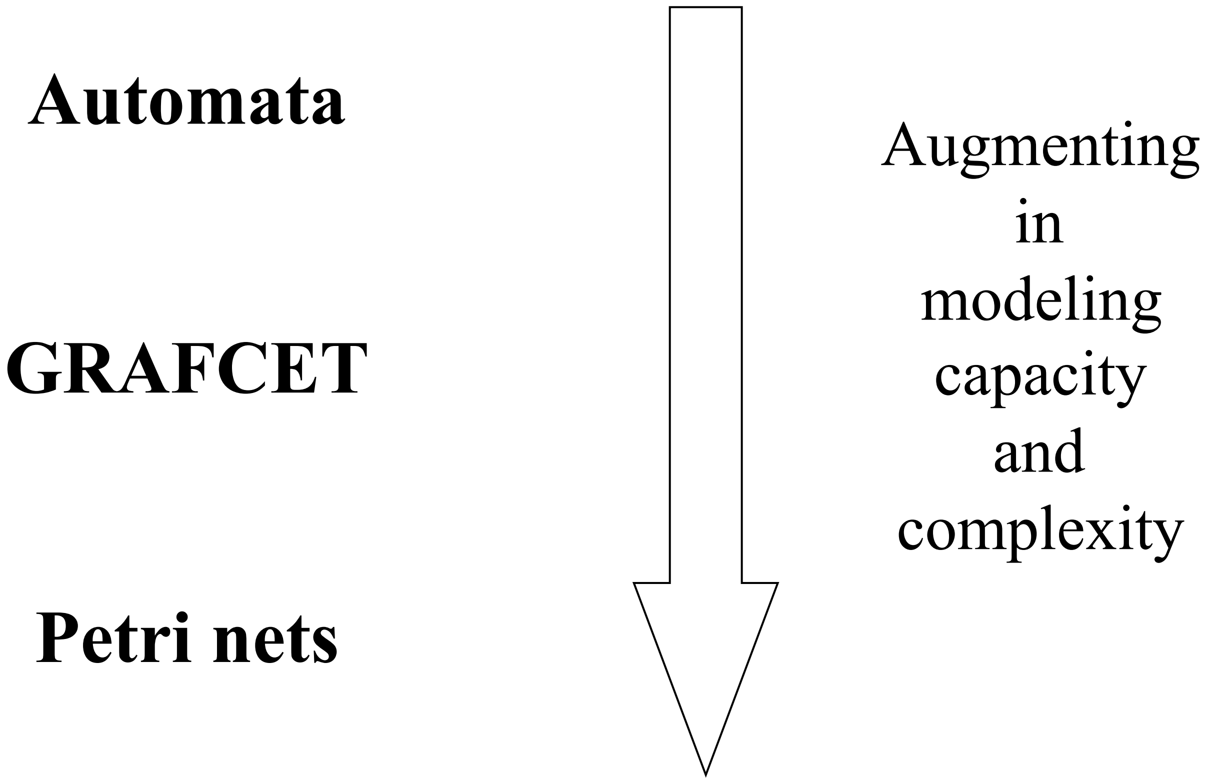
- Modeling and Analysis
- *Design* e synthesis
- Control / Supervision
- Performance assessment and robustness
- Optimization

Applications of Discrete Event Systems

- Queueing systems
- Operating systems and computers
- Telecommunications networks
- Distributed databases
- Automation

Discrete Event Systems

Typical modeling methodologies



Automata

GRAFCET

Petri nets

Augmenting
in
modeling
capacity
and
complexity

Automata Theory and Languages

Genesys of computation theory

Definition: A **language** L, defined over the alphabet **E** is a set of *strings* of finite length with events from **E**.

Exemplos: $\mathbf{E}=\{\alpha, \beta, \gamma\}$

$L_1=\{\varepsilon, \alpha\alpha, \alpha\beta, \gamma\beta\alpha\}$
L_2={all *strings* of length 3}

How to build a machine that "talks" a given language?

or

What language "talks" a system?

Properties of languages

Kleene-closure E*: set of all strings of finite length of E, including the null element ε.

Concatenation:

$$L_a L_b := \{ s \in E^* : s = s_a s_b, s_a \in L_a, s_b \in L_b \}$$

Prefix-closure:

$$\overline{L} := \{ s \in E^* : \exists_{t \in E^*} st \in L \}$$

Automata Theory and Languages

Definition: A deterministic automata is a 5-*tuple*

$$(\mathbf{E}, \mathbf{X}, \mathbf{f}, \mathbf{x_0}, \mathbf{F})$$

onde:

E	- finite alphabet (or possible events)	
X	- finite set of states	
f	- state transition function	$\mathbf{f}: \mathbf{X} \times \mathbf{E} \rightarrow \mathbf{X}$
$\mathbf{x_0}$	- initial state	$\mathbf{x_0} \subset \mathbf{X}$
F	- set of final states or marked states	$\mathbf{F} \subset \mathbf{E}$

Example of a automata

(E, X, f, x_0, F)

E = $\{\alpha, \beta, \gamma\}$

X = {x, y, z}

$\mathbf{x_0} = x$

F = {x, z}

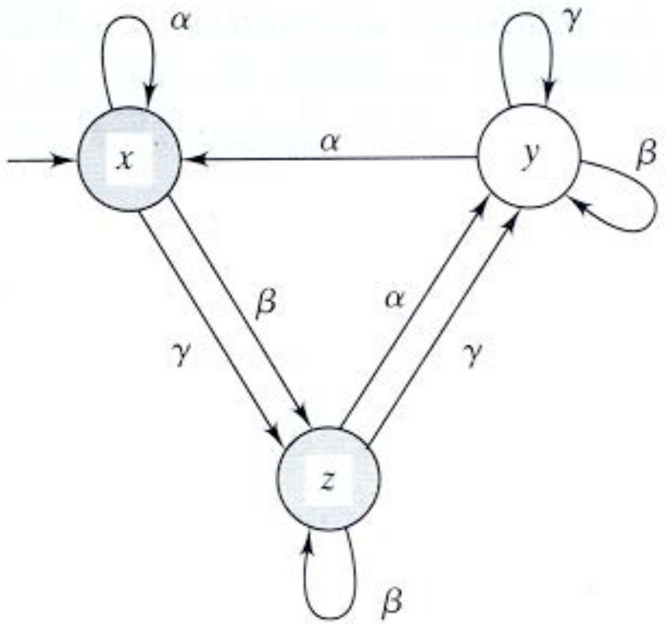


Figure 2.1. State transition diagram for Example 2.3.

$f(x, \alpha)=x$ $f(x, \beta)=z$ $f(x, \gamma)=z$

$f(y, \alpha)=x$ $f(y, \beta)=y$ $f(y, \gamma)=y$

$f(z, \alpha)=y$ $f(z, \beta)=z$ $f(z, \gamma)=y$

Example of a stochastic automata

(E, X, f, x_0, F)

$\mathbf{E} = \{\alpha, \beta\}$

$\mathbf{X} = \{0, 1\}$

$\mathbf{x_0} = 0$

$\mathbf{F} = \{0\}$

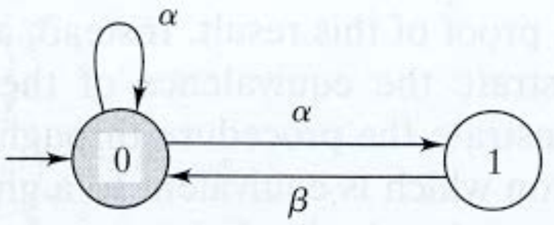


Figure 2.4. State transition diagram for the nondeterministic automaton of Example 2.7.

$f(0, \alpha) = \{0, 1\}$ $f(0, \beta) = \{\}$

$f(1, \alpha) = \{\}$ $f(1, \beta) = 0$

Given a language

$$\mathbf{G=(E, X, f, x_0 ,F)}$$

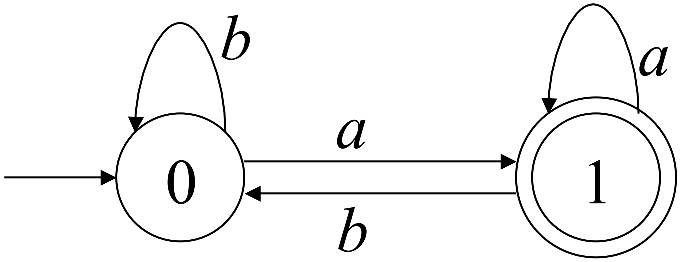
Generated language

$$\boldsymbol{L}(G) := \{s \in E^*: f(x_0,s) \text{ is defined}\}$$

Marked language

$$\boldsymbol{L_m}(G) := \{s \in E^*: f(x_0,s) \in F\}$$

Example: marked language of an automata



$$L_m(\mathrm{G}) := \{a, aa, ba, aaa, baa, bba, \ldots\}$$

Note: all strings with events *a* e *b*, followed by event *a*.

Automata equivalence:

The automata G_1 e G_2 are equivalent if

$$\boxed{\begin{gathered}\boldsymbol{L}(\mathrm{G}_1) = \boldsymbol{L}(\mathrm{G}_2) \\ \text{e} \\ \boldsymbol{L}_\mathrm{m}(\mathrm{G}_1) = \boldsymbol{L}_\mathrm{m}\,(\mathrm{G}_2)\end{gathered}}$$

Example of an automata:

Objective: To validate a sequence of events

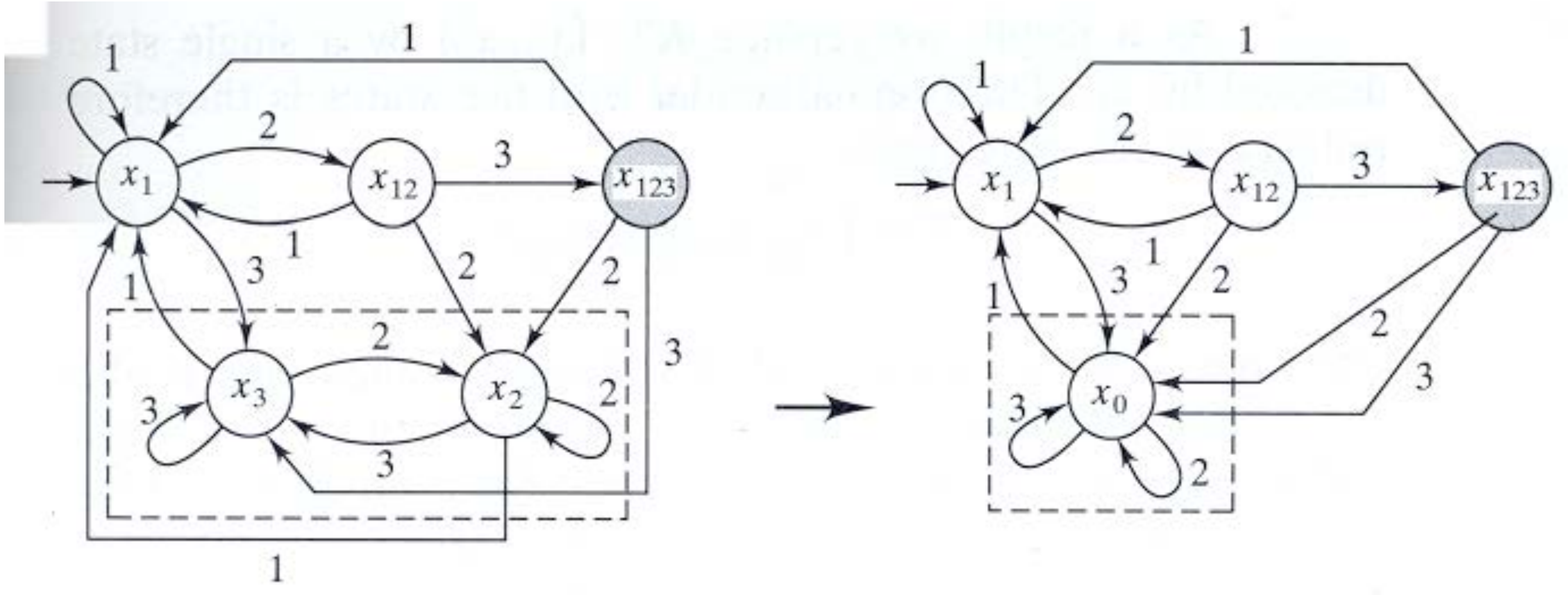
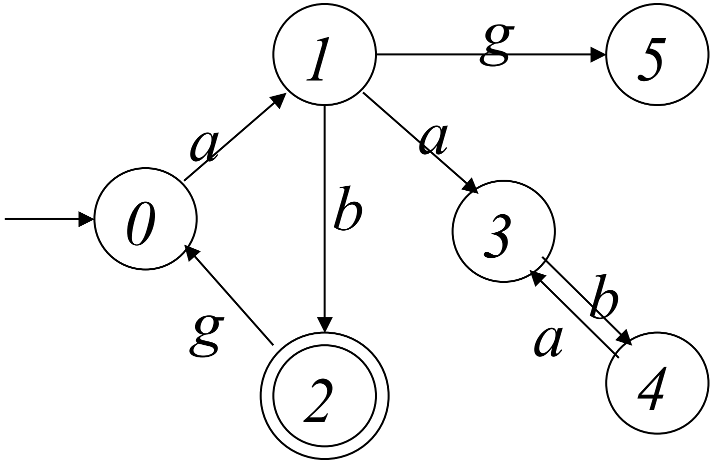


Figure 2.6. State transition diagrams for digit sequence detector in Example 2.9.

Deadlocks (*inter-blocagem*)

Example:



The state *5* is a *deadlock*.

The states *3* and *4* constitutes a *livelock*.

How to find the *deadlocks* and the *livelocks*?

Methodologies for the analysis Of Discrete Event Systems

Deadlock:

in general the following relations are verified

$$L_m(G) \subseteq \overline{L}_m(G) \subseteq L(G)$$

An automata G has a deadlock if

$$\overline{L}_m(G) \subset L(G)$$

and is not blocked when

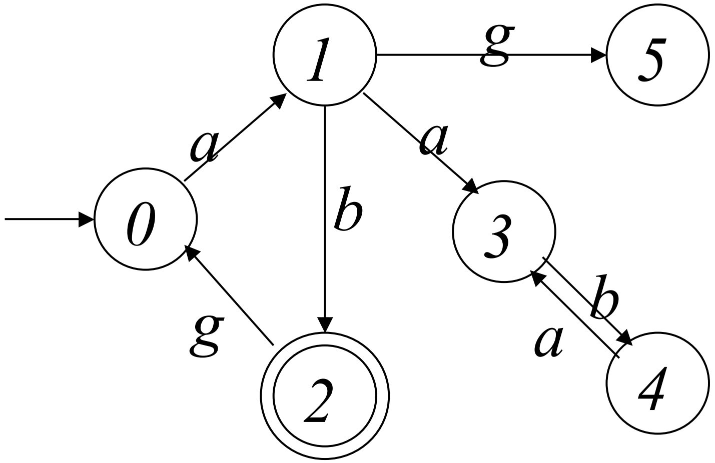
$$\overline{L}_m(G) = L(G)$$

Deadlock:

$$L_m(G) = \{ab, abgab, abgabgab, \ldots\}$$

Example:

$$L(G) = \left\{ \begin{array}{l} \varepsilon, a, ab, ag, aa, aab, \\ abg, aaba, abga, \ldots \end{array} \right\}$$



$$\left(L_m(G) \subset L(G)\right)$$

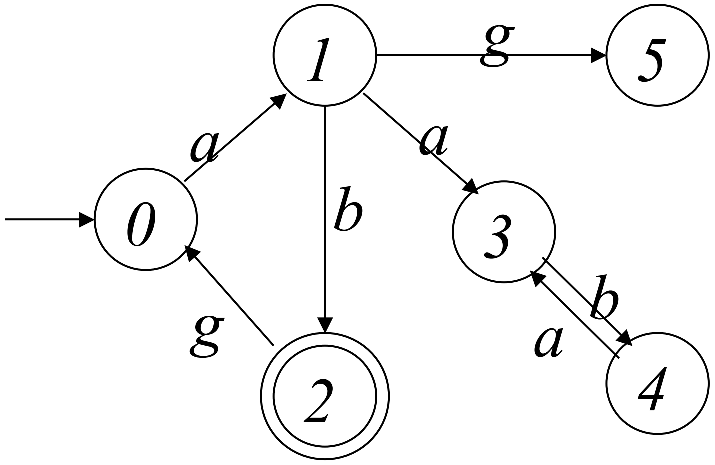
The state *5* is a *deadlock.*

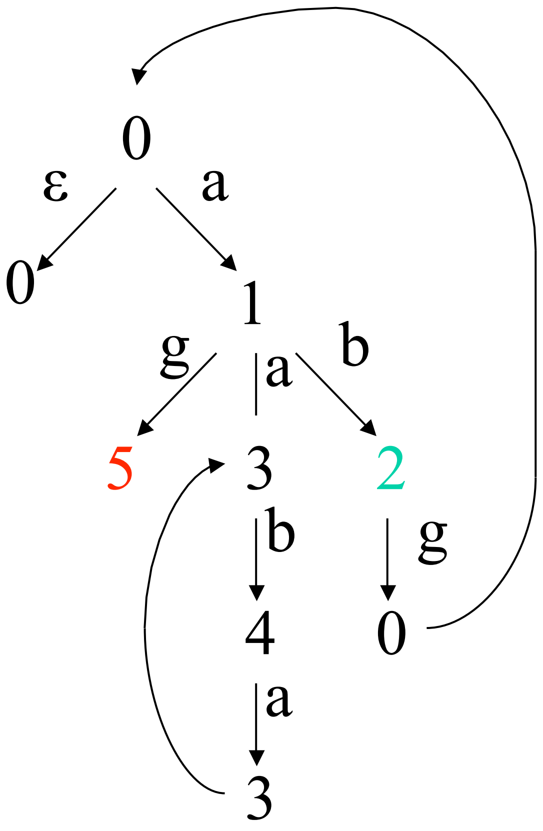
$$\overline{L}_m(G) \neq L(G)$$

The states *3* and *4* constitutes a *livelock*.

Alternative way to detect deadlocks:

Example:





The state *5* is a *deadlock.*

The states *3* and *4* constitutes a *livelock*.

Timed Discrete Event Systems

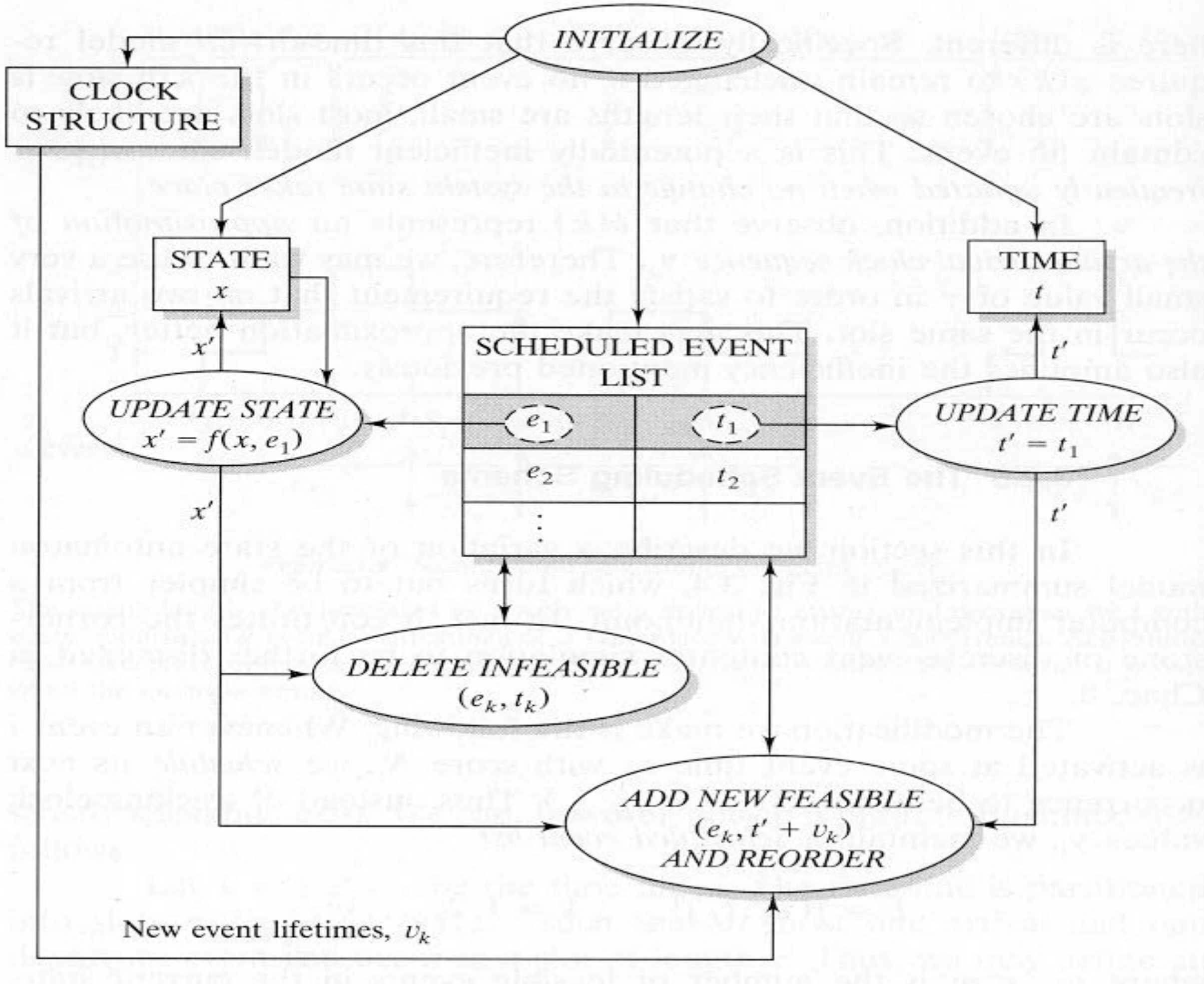


Figure 3.10. The event scheduling scheme.

Petri nets

Developed by Carl Adam Petri in his PhD thesis in 1962.

Definition: A marked Petri net is a *5-tuple*

$$\mathbf{(P, T, A, w, x_0)}$$

where:

P	- set of places	
T	- set of transitions	
A	- set of arcs	$A \subset (P \times T) \cup (T \times P)$
w	- weight function	$w: A \rightarrow \mathbb{N}$
x_0	- initial marking	$x_0 : P \rightarrow \mathbb{N}$

Example of a Petri net

(P, T, A, w, x_0)

$P=\{p_1, p_2, p_3, p_4, p_5\}$

$T=\{t_1, t_2, t_3, t_4\}$

$A=\{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_3, t_3),$
$(t_2, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_1)\}$

$w(p_1, t_1)=1, w(t_1, p_2)=1, w(t_1, p_3)=1, w(p_2, t_2)=1$
$w(p_3, t_3)=2, w(t_2, p_4)=1, w(t_3, p_5)=1, w(p_4, t_4)=3$
$w(p_5, t_4)=1, w(t_4, p_1)=1$

$x_0 = \{1, 0, 0, 2, 0\}$

Petri nets

Rules to follow (mandatory):

- An arc (directed connection) can connect places to transitions
- An arc can connect transitions to places
- A transition can have no places as inputs (source)
- A transition can have no places as outputs (sink)
- The same happens with the input and output transitions for places

Example of a Petri net

(P, T, A, w, x_0)

$P=\{p_1, p_2, p_3, p_4, p_5\}$

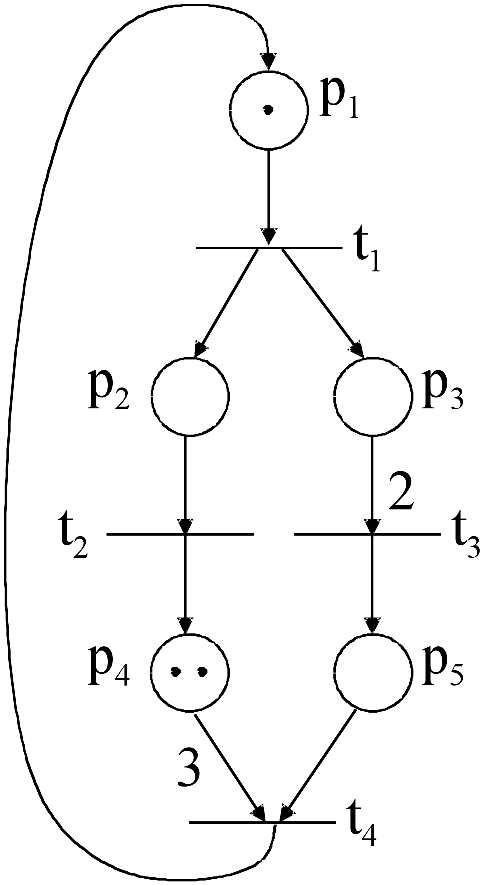
$T=\{t_1, t_2, t_3, t_4\}$

$A=\{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_3, t_3),$
$(t_2, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_1)\}$

$w(p_1, t_1)=1, w(t_1, p_2)=1, w(t_1, p_3)=1, w(p_2, t_2)=1$
$w(p_3, t_3)=2, w(t_2, p_4)=1, w(t_3, p_5)=1, w(p_4, t_4)=3$
$w(p_5, t_4)=1, w(t_4, p_1)=1$

$x_0 = \{1, 0, 0, 2, 0\}$

Petri net graph



Alternative definition of a Petri net

A marked Petri net is a *5-tuple*

$$(\mathbf{P}, \mathbf{T}, \mathbf{I}, \mathbf{O}, \mu_0)$$

where:

P	- set of places	
T	- set of transitions	
I	- transition input function	$\mathbf{I}: \mathbf{P} \rightarrow \mathbf{T}^{\infty}$
O	- transition output function	$\mathbf{O}: \mathbf{T} \rightarrow \mathbf{P}^{\infty}$
μ_0	- initial marking	$\mu_0 : \mathbf{P} \rightarrow \mathbf{N}$

Example of a Petri net and its graphical representation

Alternative definition

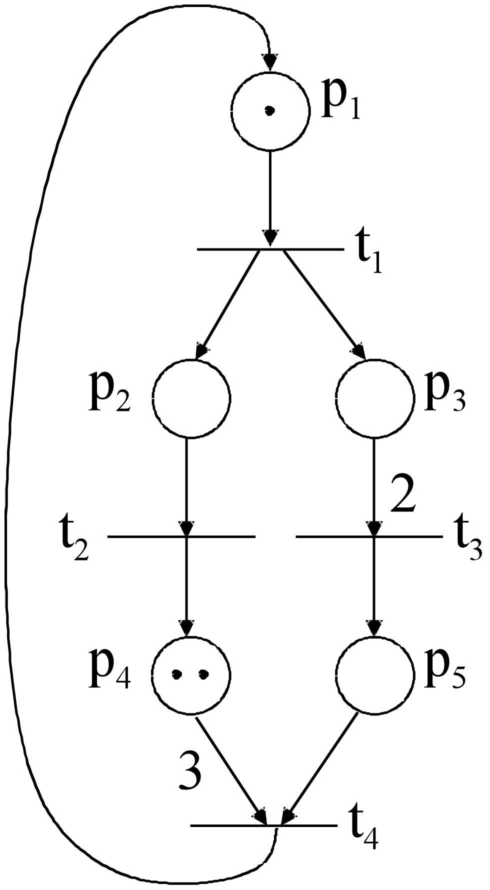
(P, T, I, O, μ_0)

$P=\{p_1, p_2, p_3, p_4, p_5\}$

$T=\{t_1, t_2, t_3, t_4\}$

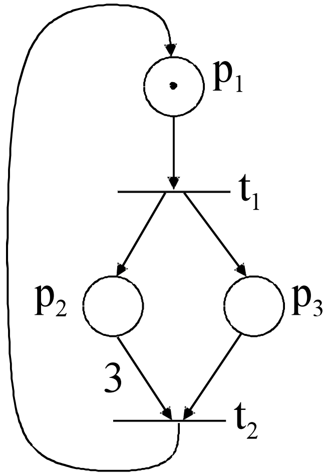
$I(t_1)=\{p_1\}$ $O(t_1)=\{p_2 , p_3\}$
$I(t_2)=\{p_2\}$ $O(t_2)=\{p_4\}$
$I(t_3)=\{p_3, p_3\}$ $O(t_3)=\{p_5\}$
$I(t_4)=\{p_4, p_4, p_4, p_5\}$ $O(t_4)=\{p_1\}$

$\mu_0 = \{1, 0, 0, 2, 0\}$



Petri nets

The state of a Petri net is characterized by the marking of all places.



The set of all possible markings of a Petri net corresponds to its state space.

How does the state of a Petri net evolves?

Dynamics of Petri nets

A transition t_j ε T is *enabled* if:

$$\forall p_i \in P: \quad \mu(p_i) \quad \geq \quad \#(p_i, I(t_j))$$

A transition t_j Î T is enabled to *fire*,
resulting in a new marking given by

$$\mu'(p_i) \; = \; \mu(p_i) \; - \; \#(p_i, I(t_j)) \; + \; \#(p_i, O(t_j))$$

Petri nets

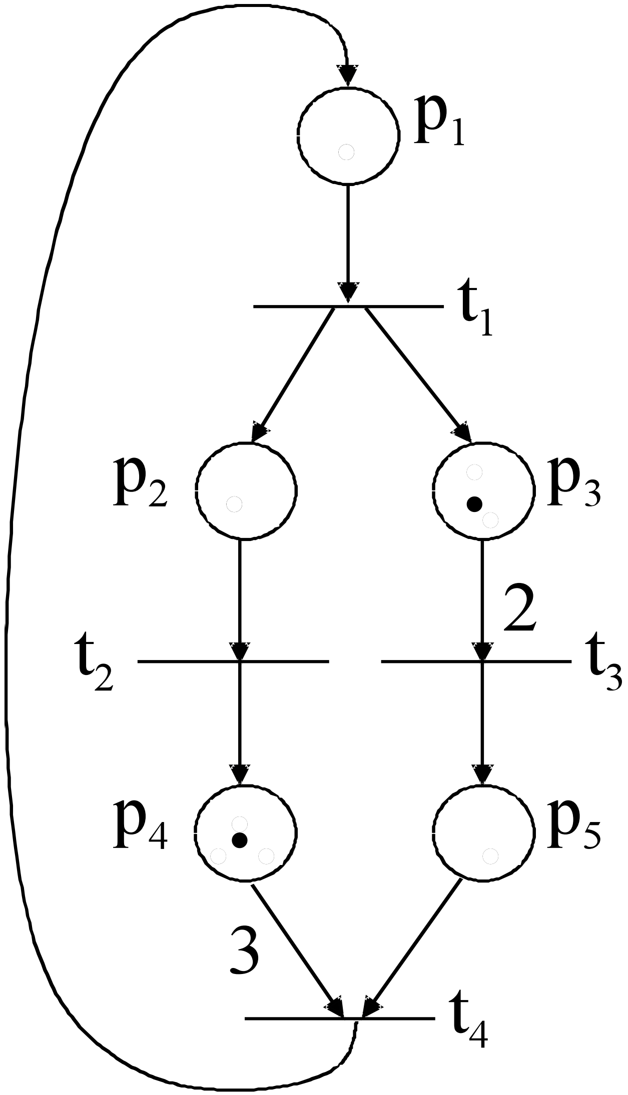
Example of evolution of a Petri net

Initial marking:

$\mu_0 = \{1, 0, 1, 2, 0\}$

This discrete event system can not change state.

It is in a *deadlock!*



Petri nets: Conditions and Events

Conditions:

a) The server is idle.
b) A job arrives and waits to be processed
c) The server is processing the job
d) The job is complete

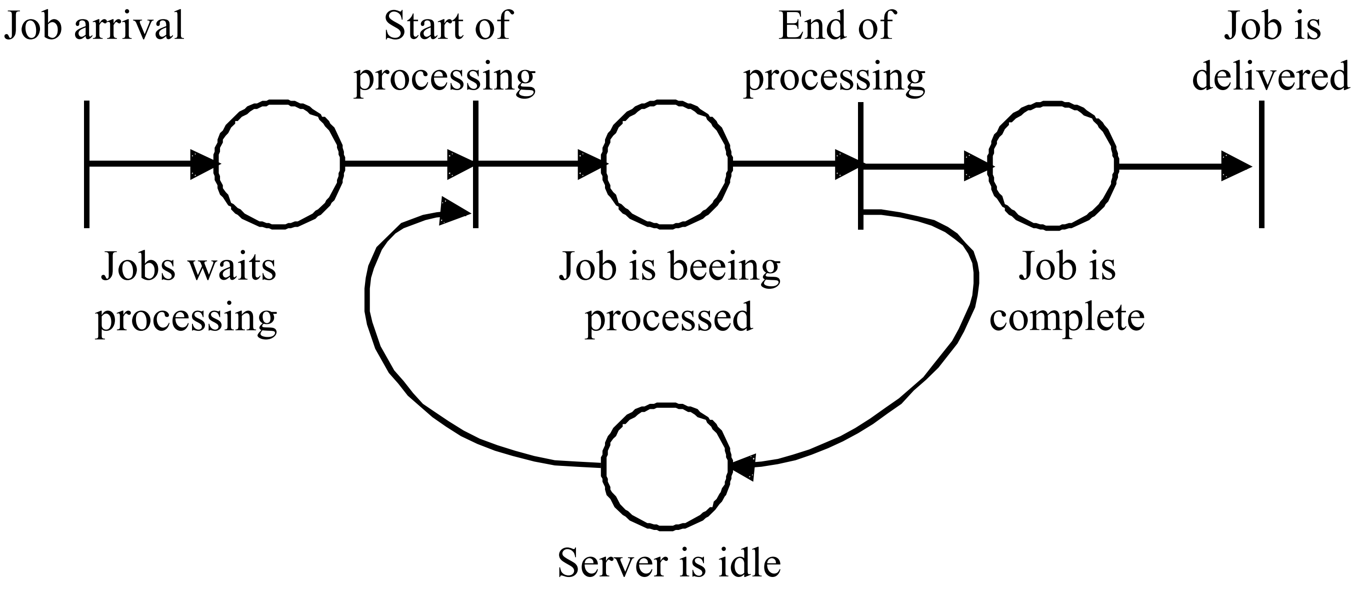
Events

1) Job arrival
2) Server starts processing
3) Server finishes processing
4) The job is delivered

Event	Pre-conditions	Pos-conditions
1	-	b
2	a,b	c
3	c	d,a
4	d	-

Petri nets: Conditions and Events

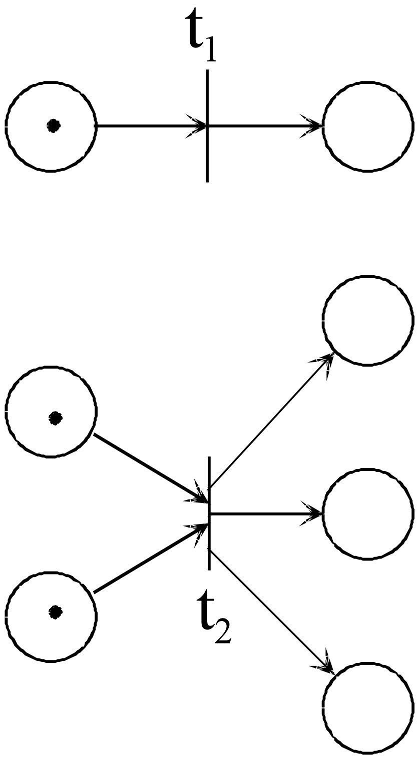
Event	Pre-conditions	Pos-conditions
1	-	b
2	a,b	c
3	c	d,a
4	d	-



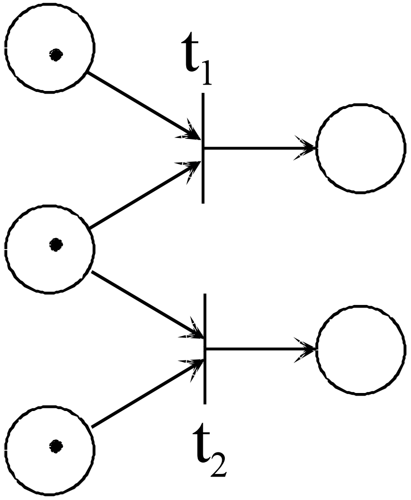
Petri nets

Modeling mechanisms

Concurrence



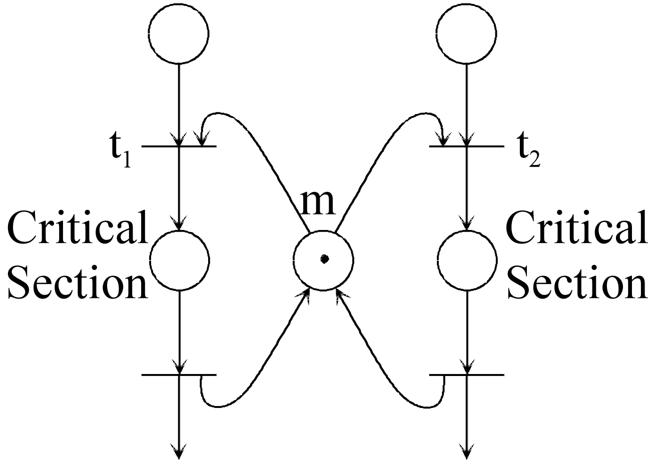
Conflict



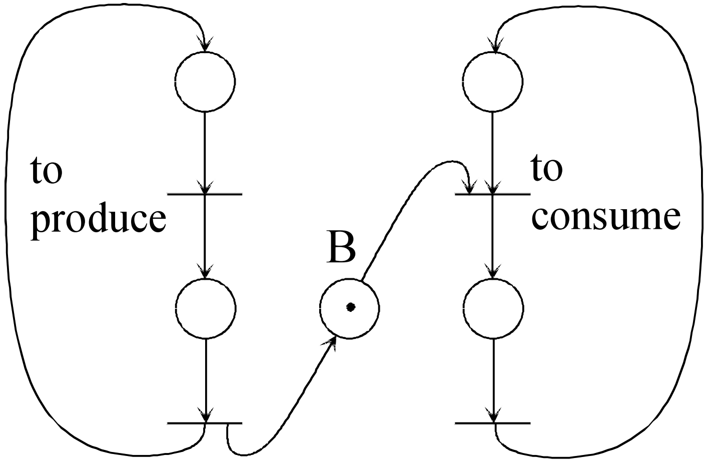
Petri nets

Modeling mechanisms

Mutual Exclusion



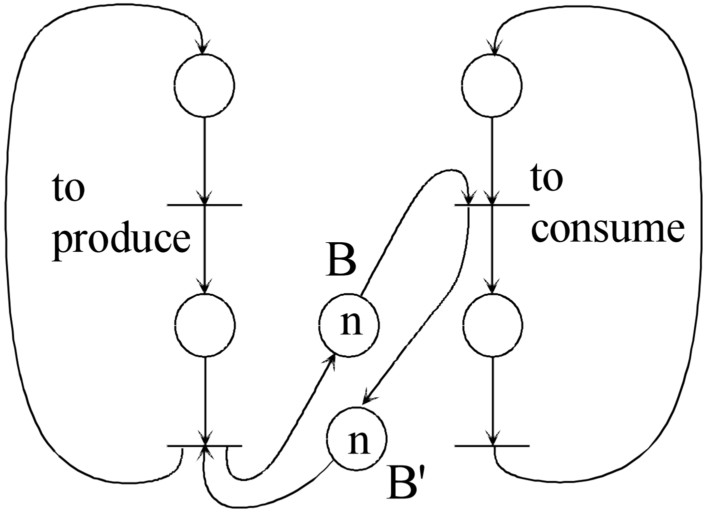
Producer / Consumer



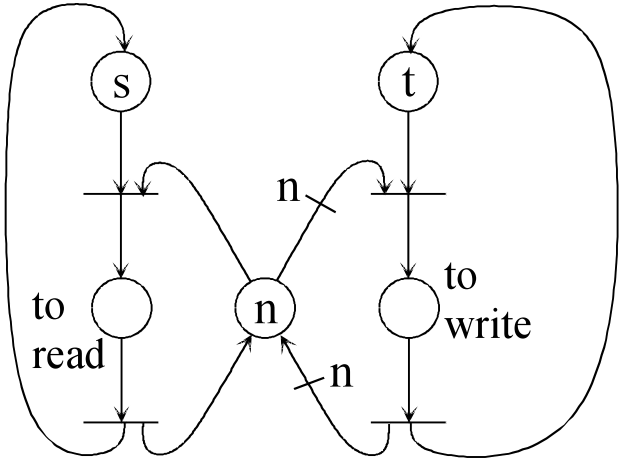
Petri nets

Modeling mechanisms

Producer / Consumer with finite capacity



Readers / Writers

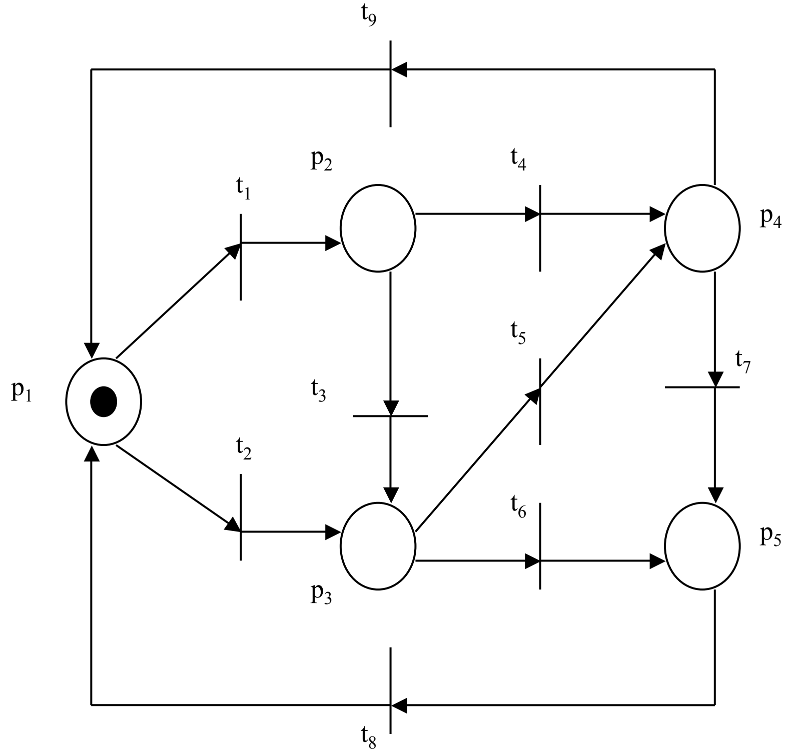


Discrete Event Systems

Example of a simple automation system modelled using PNs

An automatic soda selling machine accepts 50 c and $1 coins and sells 2 types of products: SODA A, that costs $1.50 and SODA B, that costs $2.00.

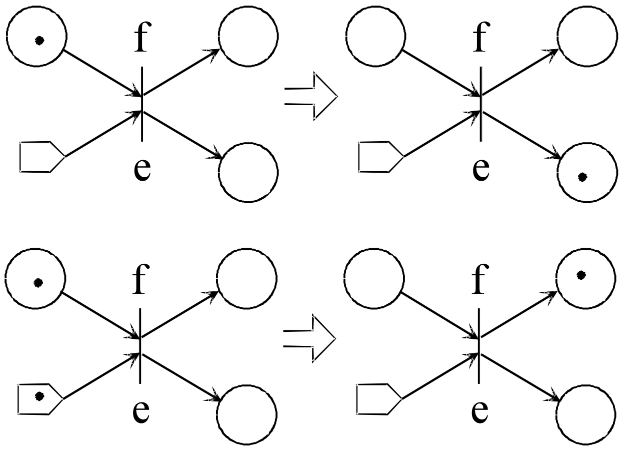
Assume that the money return operation is omitted.



p_1: machine with \$0.00;
t_1: coin of 50 c introduced;
t_8: SODA B sold.

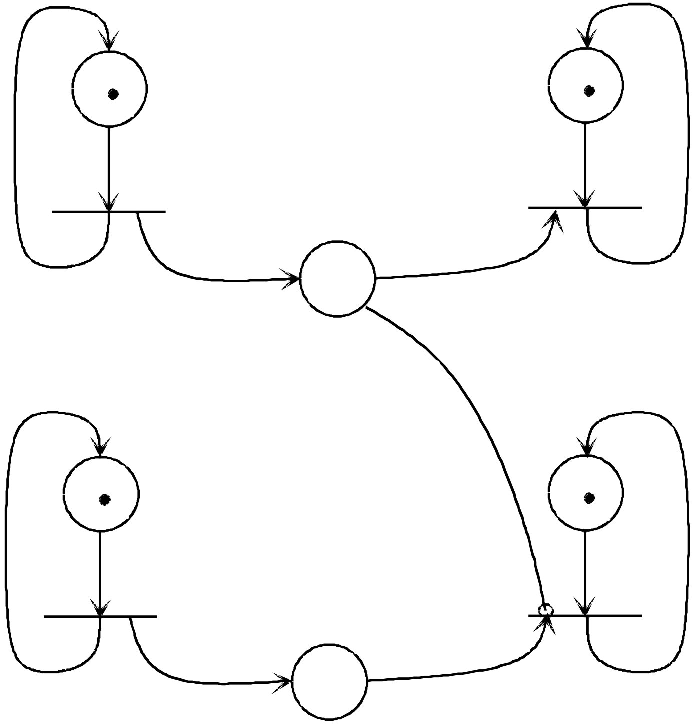
Extentions to Petri nets

Switches [Baer 1973]



Possible to be implemented with restricted Petri nets.

Extentions to Petri nets



Inhibitor Arcs

Equivalent to

nets with priorities

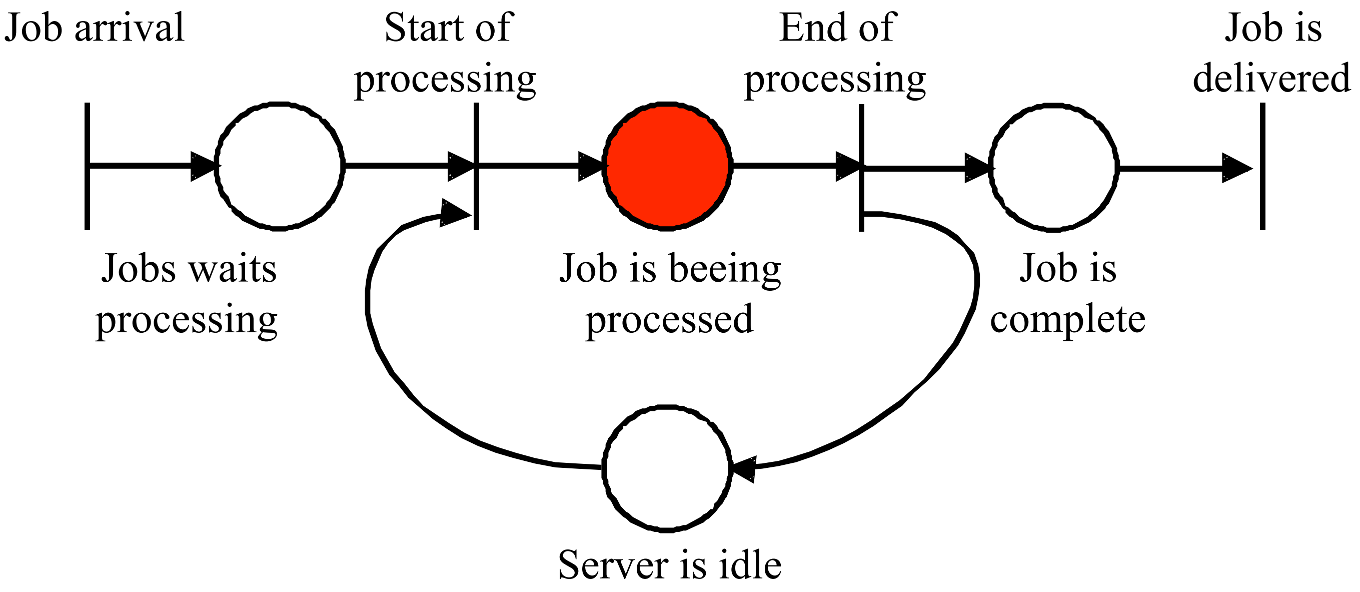
Can be implemented with restricted Petri nets?

Zero tests...

Infinity tests...

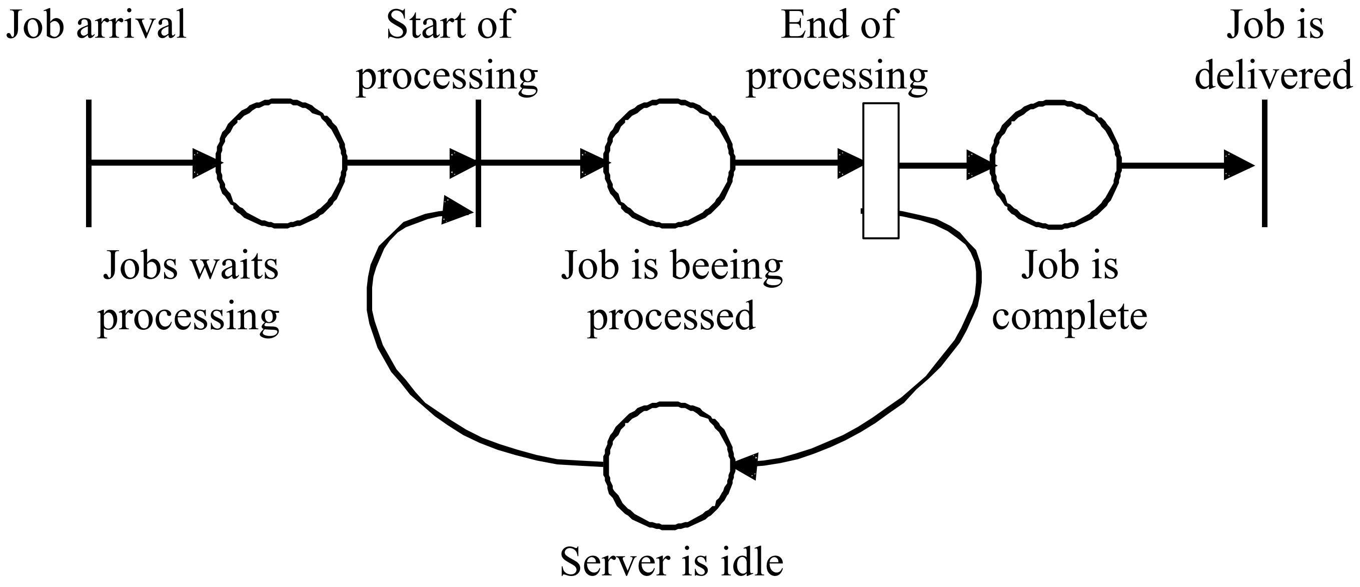
Extentions to Petri nets

P-Timed nets



Extentions to Petri nets

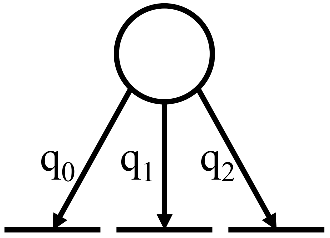
T-Timed nets



Extentions to Petri nets

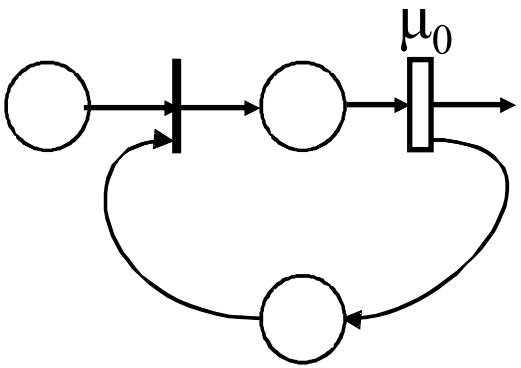
Stochastic nets

Stochastic switches



q_0+q_1+ q_2 =1

Transitions with stochastic timmings described by a stochastic variable with known pdf

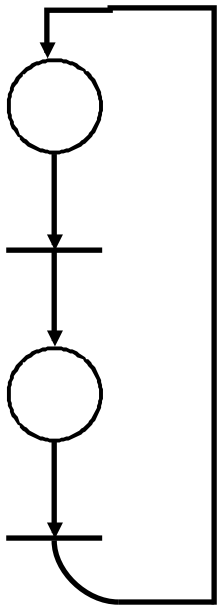


Discrete Event Systems

Sub-classes of Petri nets

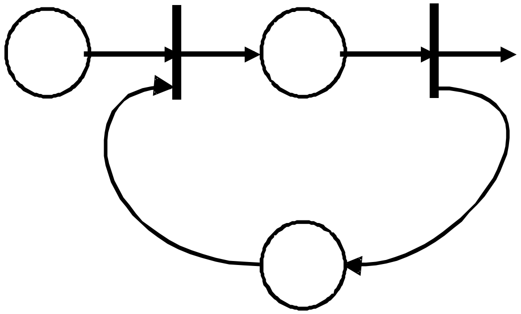
State Machine:

Petri nets where each transition has exactly one input arc and one output arc.



Marked Graphs

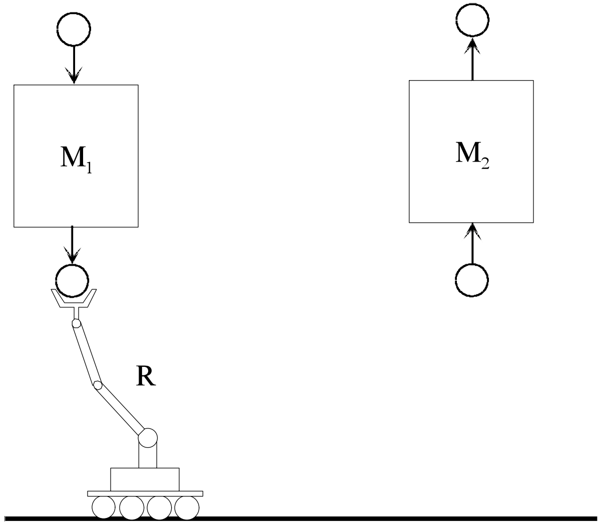
Petri nets where each place has exactly one input arc and one output arc.



Discrete Event Systems

Example of DES:

Manufacturing system composed by 2 machines (M_1 and M_2) and a robotic manipulator (R). This takes the finished parts from machine M_1 and transports them to M_2.



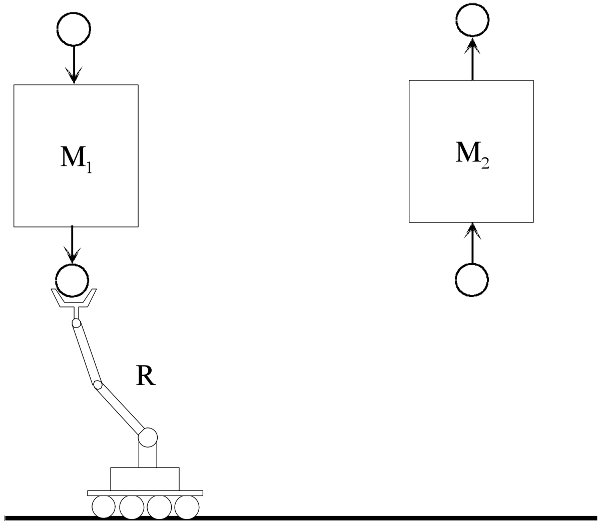
No buffers available on the machines. If R arrives near M_1 and the machine is busy, the part is rejected.

If R arrives near M_2 and the machine is busy, the manipulator must wait.

Machinning time: M_1=0.5s; M_2=1.5s; $R_{M1\,®\,M2}$=0.2s; $R_{M2\,®\,M1}$=0.1s;

Discrete Event Systems

Example of DES:



Variables of

M_1	x_1
M_2	x_2
R	x_3

Example of arrival of parts:

$$a(t)=\begin{cases} 1 \quad em \quad \{0.1,\ \ 0.7,\ \ 1.1,\ \ 1.6,\ \ 2.5\} \\ 0 \quad em \quad todos \quad os \quad outros \quad ins\,tan\,tes \end{cases}$$

x_1={Idle, Busy, Waiting}

x_2={Idle, Busy}

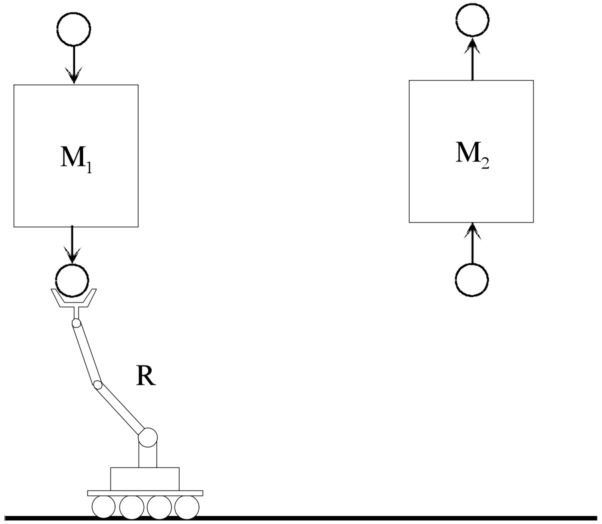
x_3={Idle, Carrying, Returning}

Discrete Event Systems

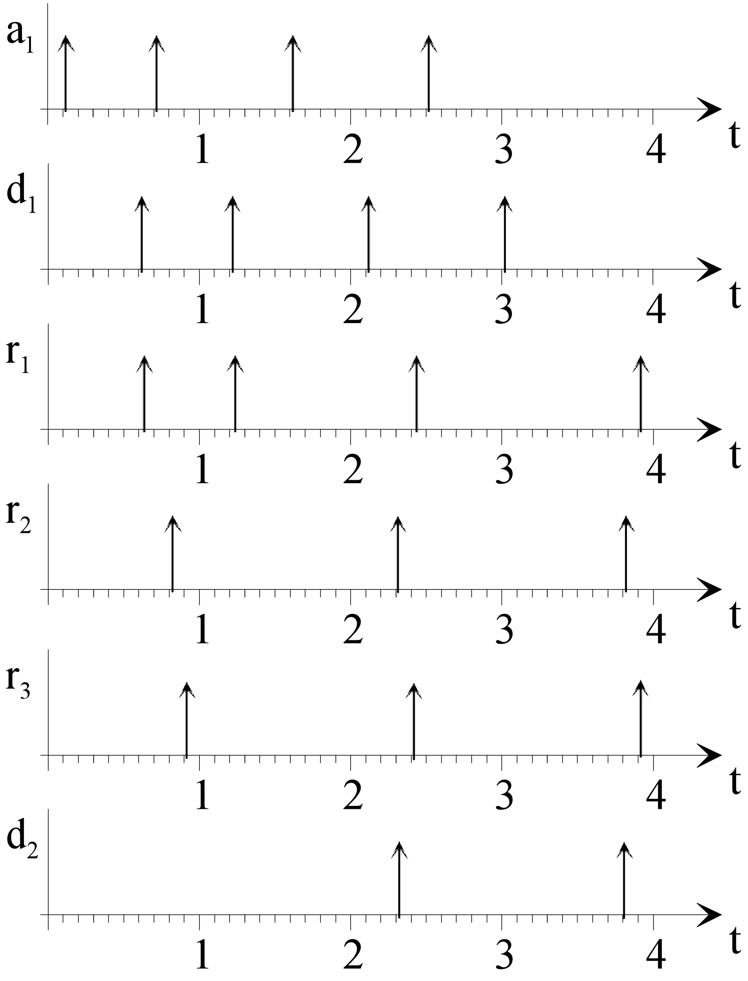
Example of DES:

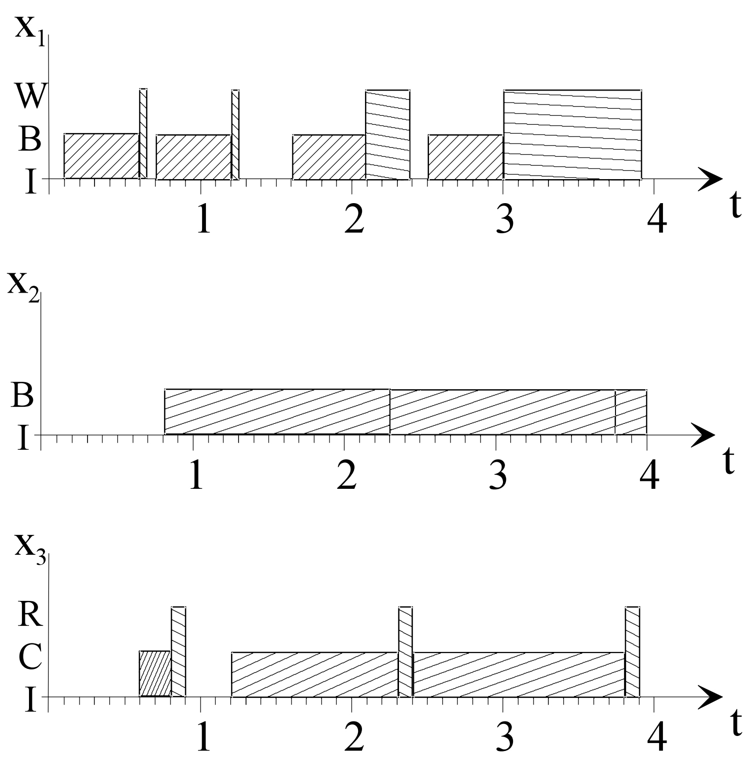
Definition of events:

a_1	- loads part in M_1
d_1	- ends part processing in M_1
r_1	- loads manipulator
r_2	- unloads manipulator and loads M_2
d_2	- ends part processing in M_2
r_3	- manipulator at base



Discrete Event Systems



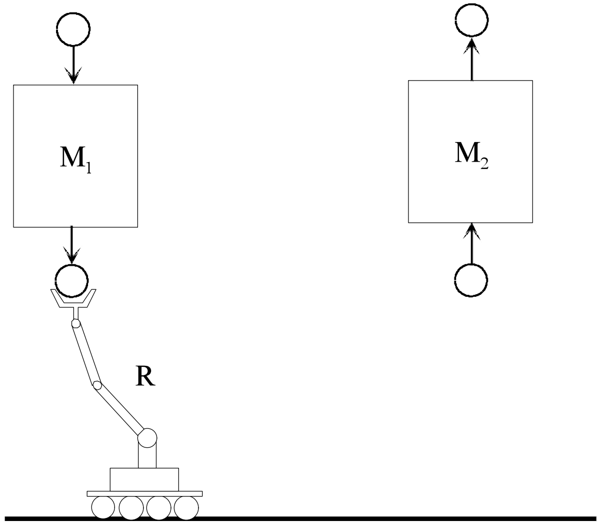


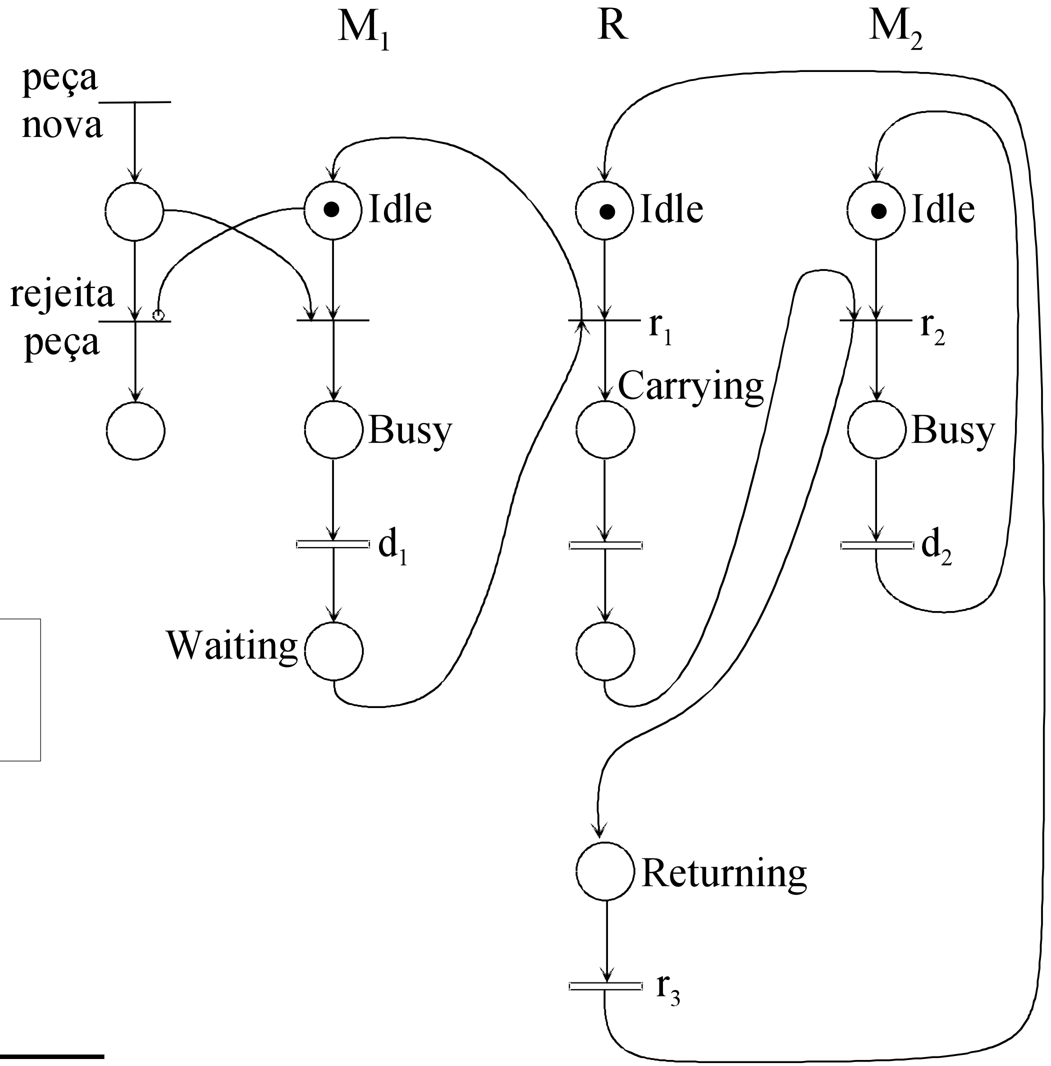
Discrete Event Systems

Example of DES:

Events:

a_1 - loads part in M_1
d_1 - ends part processing in M_1
r_1- loads manipulator
r_2- unloads manipulator and loads M_2
d_2- ends part processing in M_2
r_3- manipulator at base





Industrial Automation

(Automação de Processos Industriais)

Analysis of Discrete Event Systems

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

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

Syllabus:

Chap. 6 – Discrete Event Systems [2 weeks]

…

Chap. 7 – Analysis of Discrete Event Systems [2 weeks]

Properties of DESs.

Methodologies to analyze DESs:

* The Reachability tree.
* The Method of Matrix Equations.

…

Chap. 8 – DESs and Industrial Automation [1 week]

Some pointers to Sistemas de Eventos Discretos

History: http://prosys.changwon.ac.kr/docs/petrinet/1.htm

Tutorial: http://vita.bu.edu/cgc/MIDEDS/
http://www.daimi.au.dk/PetriNets/

Analyzers, and Simulators: http://www.ppgia.pucpr.br/~maziero/petri/arp.html (in Portuguese)
http://wiki.daimi.au.dk:8000/cpntools/cpntools.wiki
http://www.informatik.hu-berlin.de/top/pnk/download.html

Bibliography:

* Cassandras, Christos G., "Discrete Event Systems - Modeling and PerformanceAnalysis", Aksen Associates, 1993.

* Peterson, James L., "Petri Net Theory and the Modeling of Systems", Prentice-Hall,1981. Online em http://prosys.changwon.ac.kr/docs/petrinet/

* Petri Nets and GRAFCET: Tools for Modelling Discrete Event Systems R. DAVID, H. ALLA, New York : PRENTICE HALL Editions, 1992

Properties of Discrete Event Systems

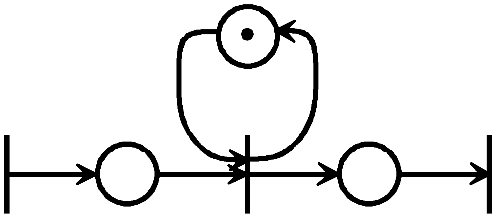
Reachability

Given a Petri net C=(P, T, I, O, μ_0) with initial marking μ_0 , the set of all markings that can be obtained is the reachable set $\mu' \in R(C,\mu)$.

Note: in general is infinite!

How to compute $R(C,\mu)$?

How to describe $R(C,\mu)$?



Properties of Discrete Event Systems

Coverability

Given a Petri net C=(P, T, I, O, μ_0) with initial marking μ_0 , the state $\mu' \in R(C,\mu)$ is covered if $\mu'(i) \leq \mu(i)$, for all places $p_i \in P$.

Is it possible to use this property to help

on the search for the reachable set?

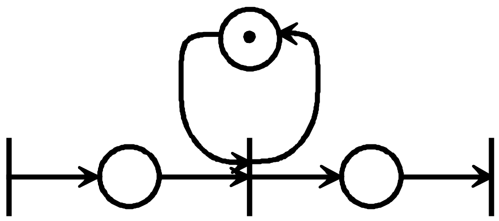
Yes!, see next...

Properties of Discrete Event Systems

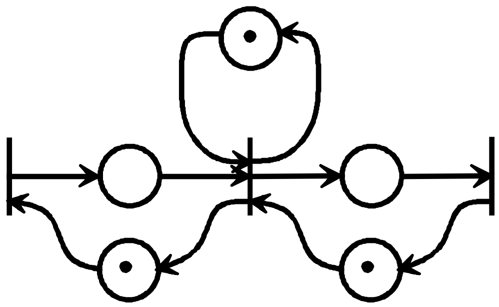
Safeness

A place $p_i \in P$ of the Petri net $C=(P, T, I, O, \mu_0)$ is safe if for all $\mu' \in R(C,\mu_0)$: $\mu_i' \leq 1$.

A Petri net is safe if all its places are safe.



Petri net not safe



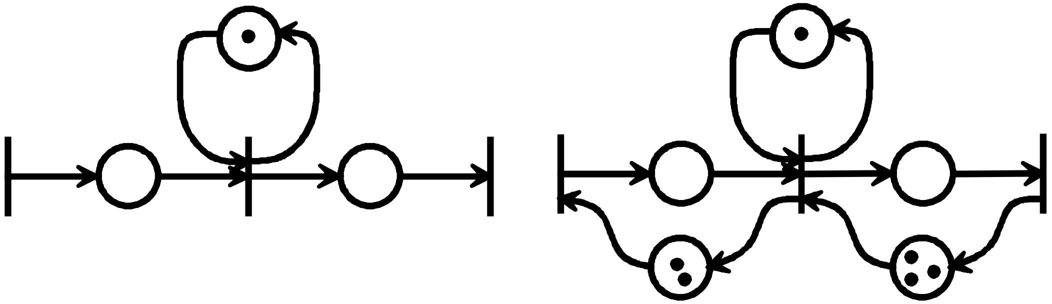
Petri net safe

Properties of Discrete Event Systems

Boundness

A place $p_i \in P$ of the Petri net C=(P, T, I, O, μ_0) is k-bounded if for all $\mu' \in R(C,\mu_0)$: $\mu_i' \leq k$.

A Petri net is k-bounded if all places are k-bounded.



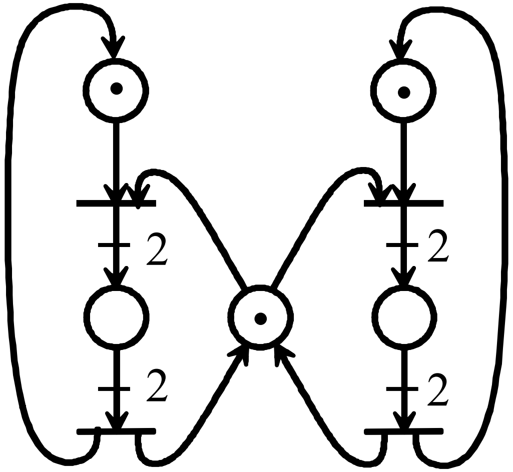
Petri net not bounded Petri net 3-bounded

Properties of Discrete Event Systems

Conservation

A Petri net C=(P, T, I, O, μ_0) is **strictly conservative** if for all μ’ ∈ R(C, μ)

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i)$$



Petri net strictly conservative

Properties of Discrete Event Systems

Liveness

A transition t_j is live of

Level 0 - if it can never be fired.

Level 1 - if it is potentially firable, that is if there exists $\mu' \in R(C,\mu)$ **such that** t_j **is enabled in** μ'.

Level 2 - if for each integer *n*, there exists a firing sequence such that t_j **occurs** n **times.**

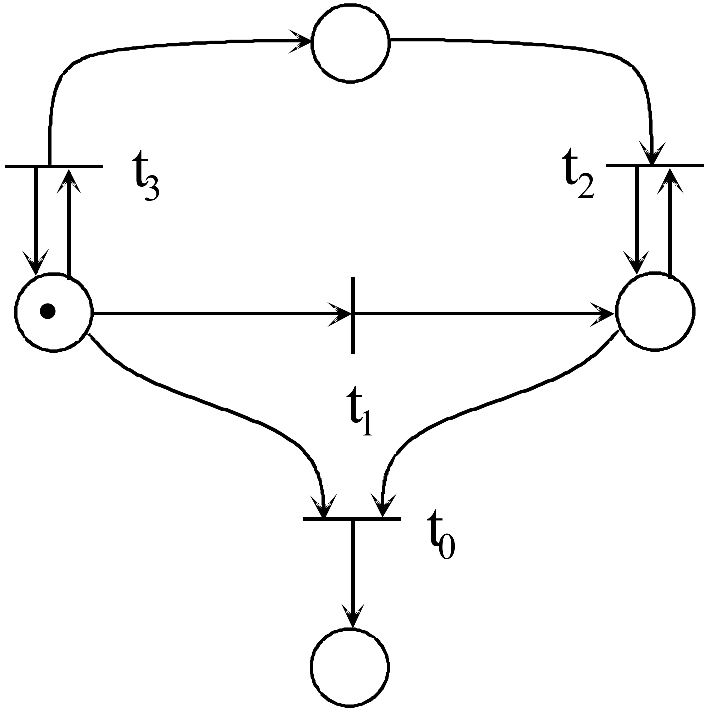
Level 3 - if there exists an infinite firing sequence such that t_j **occurs infinite times.**

Level 4 - if for each $\mu' \in R(C,\mu)$ **there exist a sequence** s **such that the transition** t_j **is enabled.**

Properties of Discrete Event Systems

Example of liveness of transitions

- t_0 is of level 0.
- t_1 is of level 1.
- t_2 is of level 2.
- t_3 is of level 3.



Properties of Discrete Event Systems

Reachability

Given a Petri net C=(P, T, I, O, μ_0) with initial marking μ_0 and a marking μ', is $\mu' \in R(C, \mu_0)$ reachable?

Analysis methods:

- Brute force...
- Reachability tree
- Matrix Equations

Analysis Methods

Reachability Tree

Build the tree of reachable markings;
Constituted by three types of nodes:

- terminals
- interiors
- duplicated

This method can also be used to study the other properties previously introduced.

See examples...

The infinity marking symbol (ω) is introduced whenever a marking covers other. Used to allow to obtain finite trees.

Analysis Methods

Reachability Tree

Algebra of the infinity symbol (ω):

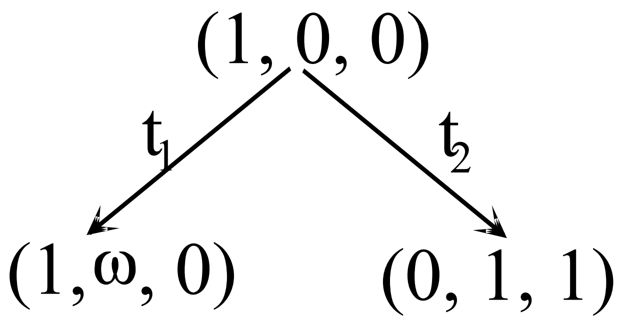
For every positive integer *a* the following relations are verified:

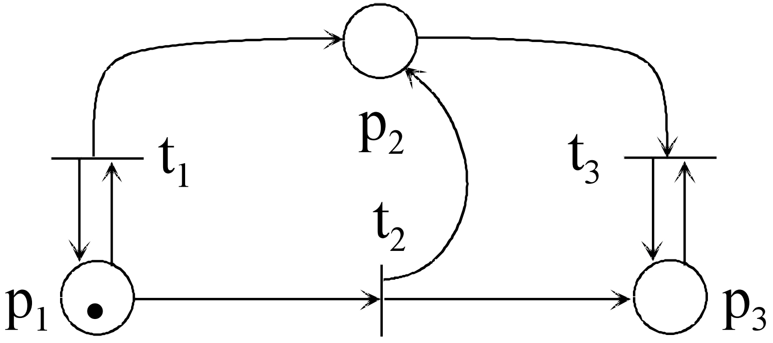
1. $\omega + a = \omega$
2. $\omega - a = \omega$
3. $a < \omega$
4. $\omega \leq \omega$

Theorem - If there exist terminal nodes in the reachability tree then the corresponding Petri net has *deadlocks*.

Analysis Methods

Example of reachability tree:



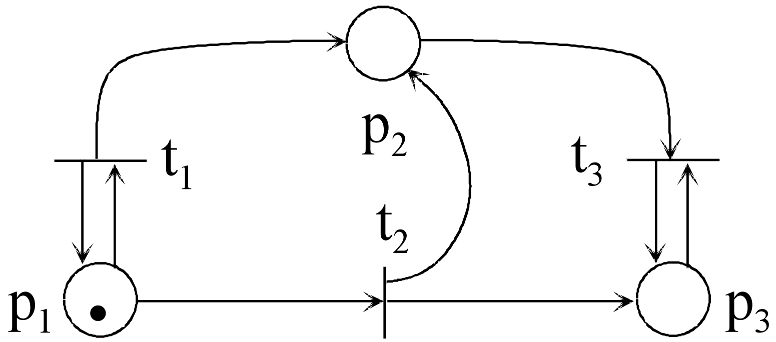


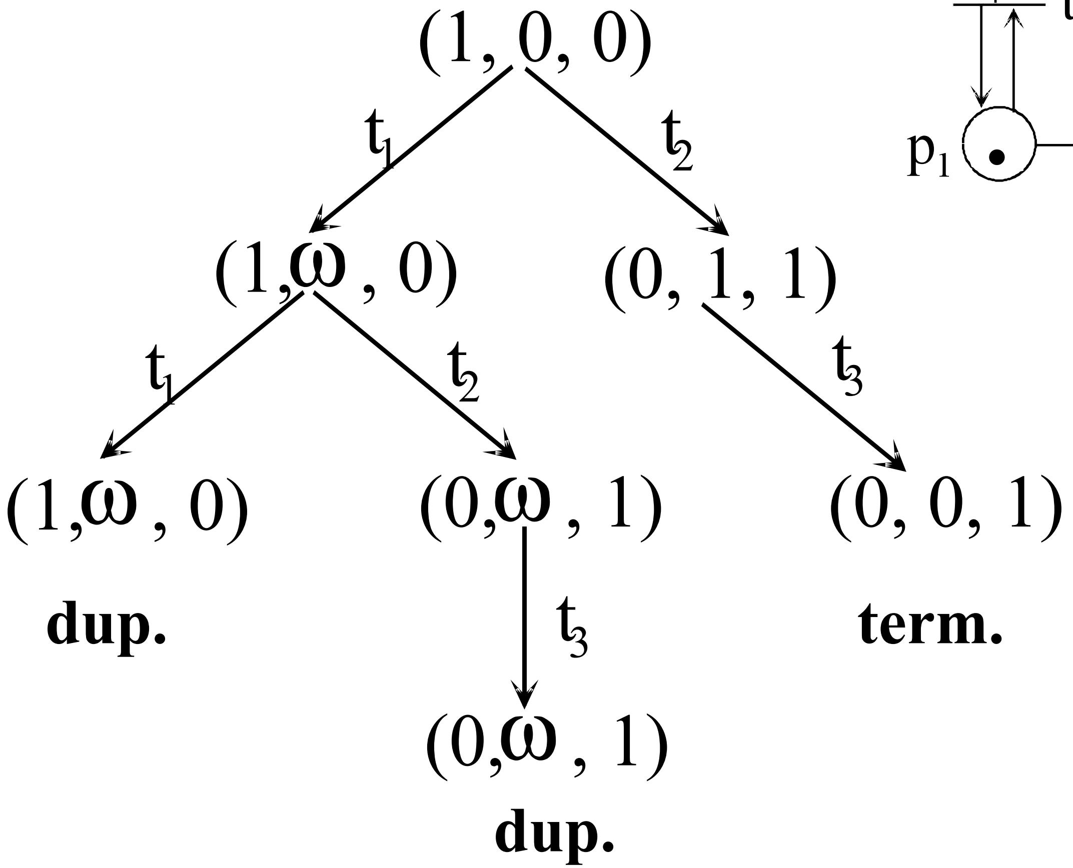
... but (1, 1, 0) covers (1, 0, 0)!

Then the infinity symbol ω can be introduced.

Analysis Methods

Example of reachability tree:





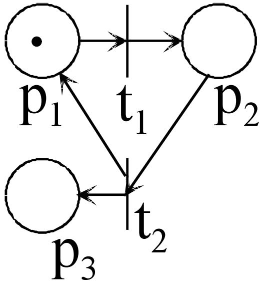
We can conclude imediately that there are

DEADLOCKS!

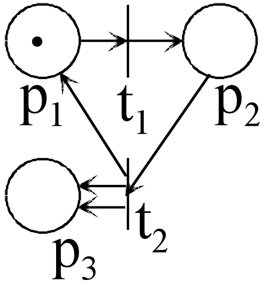
Other example:

(or a couter-example)

Different reachable sets with the same reachability tree!!!



$$\begin{tikzcd}
(1, 0, 0) \arrow[d, "t_1"] \\
(0, 1, 0) \arrow[d, "t_2"] \\
(1, 0, \omega) \arrow[d, "t_1"] \\
(0, 1, \omega) \arrow[d, "t_2"] \\
(1, 0, \omega)
\end{tikzcd}$$



$$\begin{tikzcd}
(1, 0, 0) \arrow[d, "t_1"] \\
(0, 1, 0) \arrow[d, "t_2"] \\
(1, 0, \omega) \arrow[d, "t_1"] \\
(0, 1, \omega) \arrow[d, "t_2"] \\
(1, 0, \omega)
\end{tikzcd}$$

Decidibility

Problem

Example of a Petri net

(P, T, A, w, x_0)

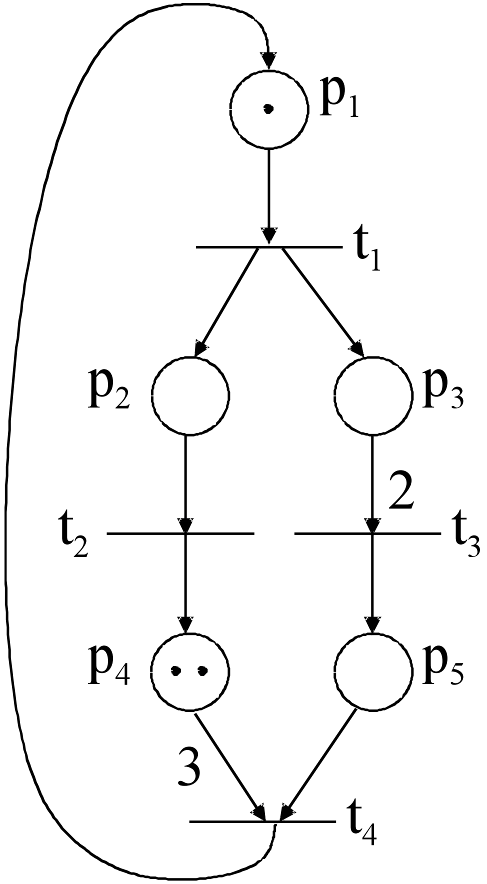
$P=\{p_1, p_2, p_3, p_4, p_5\}$

$T=\{t_1, t_2, t_3, t_4\}$

$A=\{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_3, t_3),$
$(t_2, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_1)\}$

$w(p_1, t_1)=1, w(t_1, p_2)=1, w(t_1, p_3)=1, w(p_2, t_2)=1$
$w(p_3, t_3)=2, w(t_2, p_4)=1, w(t_3, p_5)=1, w(p_4, t_4)=3$
$w(p_5, t_4)=1, w(t_4, p_1)=1$

$x_0 = \{1, 0, 0, 2, 0\}$

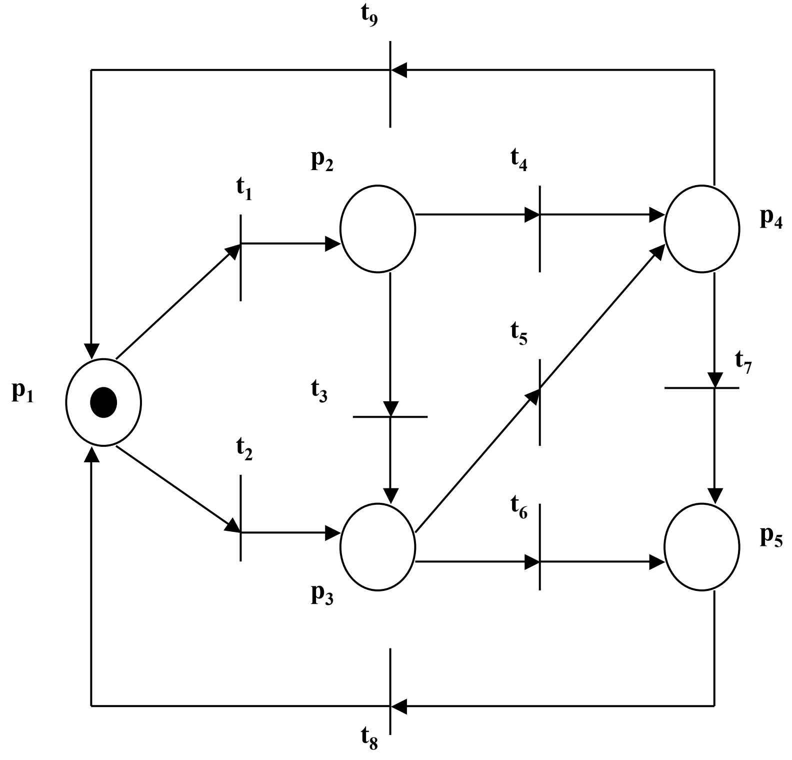


Discrete Event Systems

Example of a simple automation system modelled using PNs

An automatic soda selling machine accepts 50 c and $1 coins and sells 2 types of products: SODA A, that costs $1.50 and SODA B, that costs $2.00.

Assume that the money return operation is omitted.



p_1: machine with $0.00;
t_1: coin of 50 c introduced;
t_8: SODA B sold.

Analysis Methods

Method of the Matrix Equations (of State Evolution)

The dynamics of the Petri net state can be written in compact form as:

$$\mu(k+1) = \mu(k) + Dq(k)$$

This method can also be used to study the other properties previously introduced.

Requires some thought... ;)

where:

$\mu(k+1)$	- marking to be reached
$\mu(k)$	- initial marking
$q(k)$	- firing vector (transitions)
D	- incidence matrix. Accounts the balance of tokens, giving the transitions fired.

Analysis Methods

How to build the Incidence Matrix?

For a Petri net with *n* places and *m* transitions

$$\mu \in N_0^{\;n}$$

$$q \in N_0^{\;m}$$

$$D = D^+ - D^- \qquad \in \mathrm{Z}^{n\times m}$$

The enabling firing rule is $\mu \geq D^- q.$

Can also be writen in compact form as the inequality

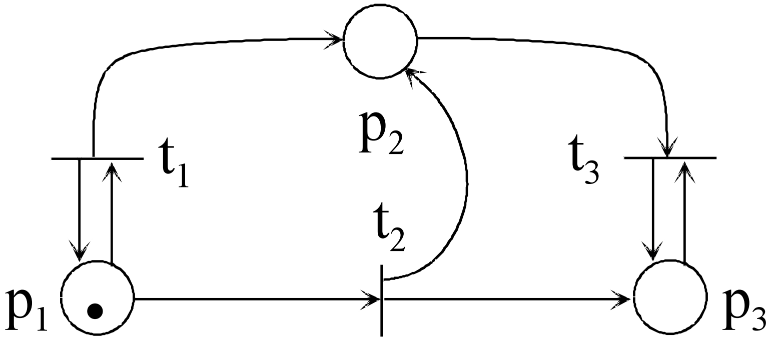
$$\mu + Dq \geq 0,$$

interpreted element by element.

Analysis Methods

Example on the use of the method of matrix equations

$$\mu(k+1) = \mu(k) + Dq(k)$$



$$\mu(k+1) = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}, \ \mu(k) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 1 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad q(k) = \begin{bmatrix} \sigma_{t1} \\ \sigma_{t2} \\ \sigma_{t3} \end{bmatrix} \quad \begin{cases} 1 = 1 - \sigma_{t2} \\ 3 = \sigma_{t1} + \sigma_{t2} - \sigma_{t3} \\ 0 = \sigma_{t2} \end{cases} \quad \begin{cases} \sigma_{t2} = 0 \\ \sigma_{t1} - \sigma_{t3} = 3 \end{cases}$$

Verify!

Analysis Methods

Properties that can be studied immediately with the Method of Matrix Equations

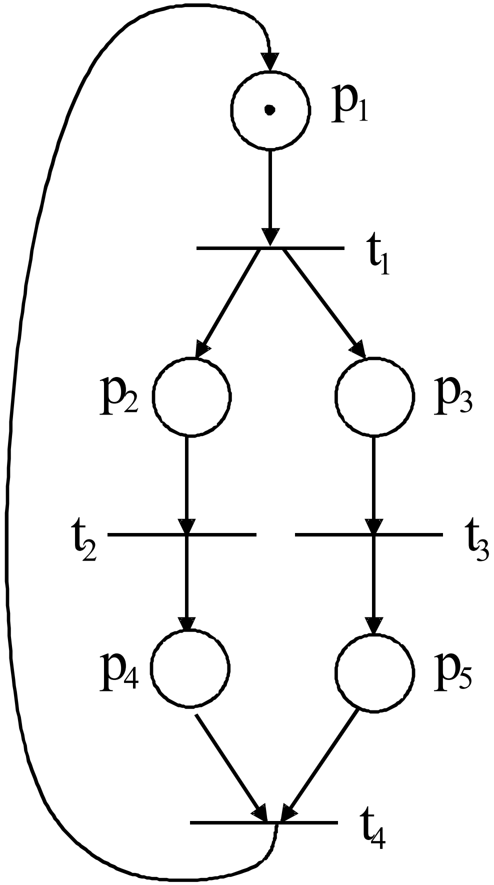
- Reachability (sufficient condition)

Theorem – if the problem of finding the transition firing vector that drives the state of a Petri net from μ to state μ' has no solution, resorting to the method of matrix equations, then the problem of reachability of μ' does not have solution.

- Conservation – the firing vector is a by-product of the MME.

- Temporal invariance – cycles of operation can be found.

Example of a Petri net



Conservation

For the number of tokens (weighted) to be preserved :

$$x^T \mu' = x^T \mu + x^T Dq$$

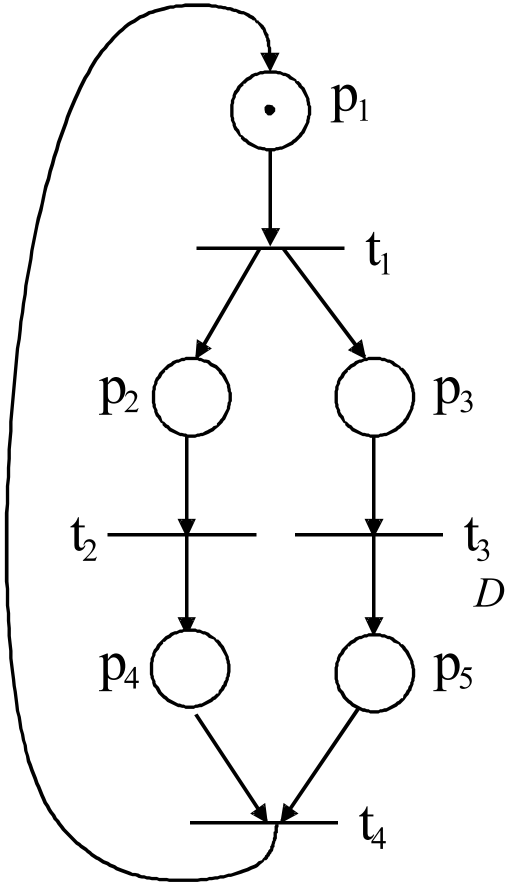
$$x^T D = 0$$

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \left\{ \begin{array}{c} -x_1 + x_2 + x_3 = 0 \\ -x_2 + x_4 = 0 \\ -x_3 + x_5 = 0 \\ x_1 - x_4 - x_5 = 0 \end{array} \right. \left\{ \begin{array}{c} x_1 = x_2 + x_3 \\ x_2 = x_4 \\ x_3 = x_5 \end{array} \right.$$

Solution: undetermined system of equations $x^T = [2 \quad 1 \quad 1 \quad 1 \quad 1].$

Example of a Petri net

Temporal invariance



To determine the transition firing vectors that make the Petri net return to the same state(s)

$$Dq = 0$$

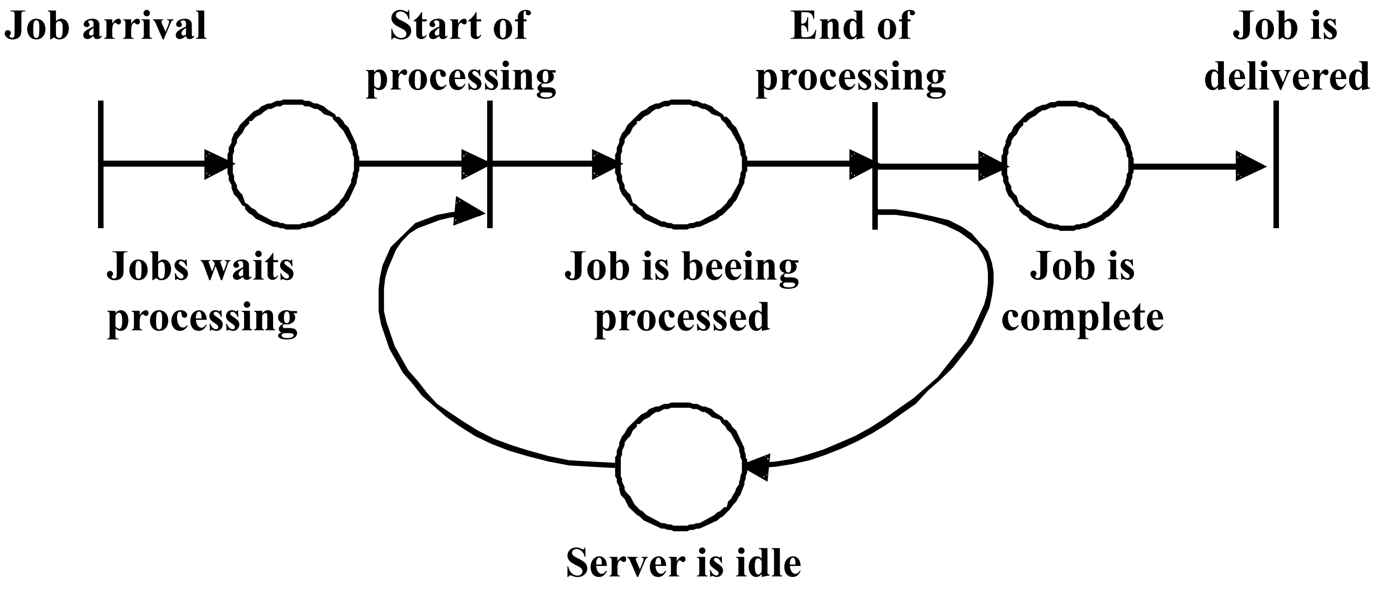
$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \quad q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad \begin{cases} -q_1 + q_4 = 0 \\ q_1 - q_2 = 0 \\ q_1 - q_3 = 0 \\ q_2 - q_4 = 0 \\ q_3 - q_4 = 0 \end{cases}$$

$$q = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Solution: undetermined system of equations

Example for the analysis of properties:

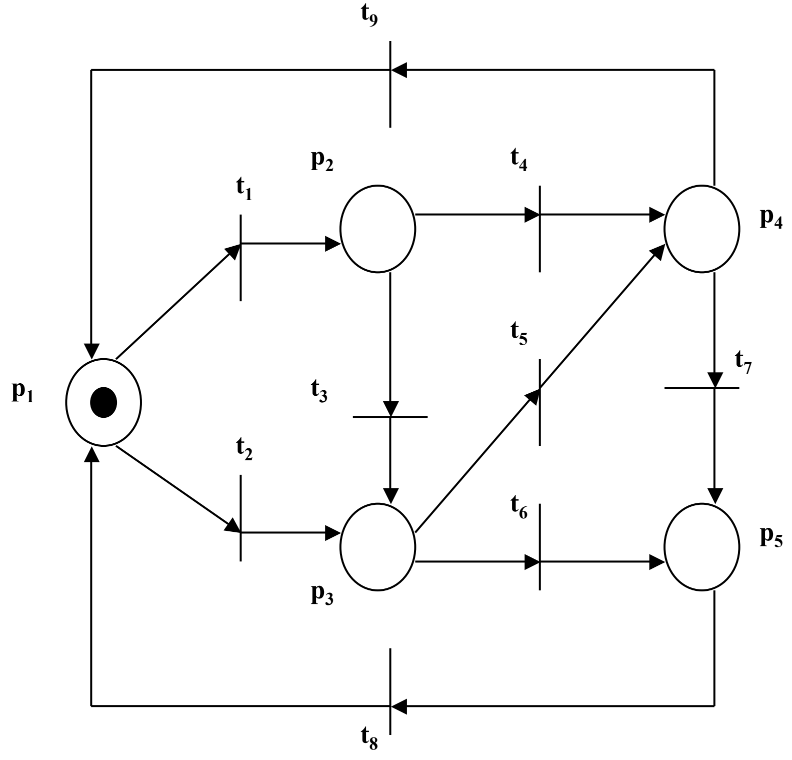
Event	Pre-conditions	Pos-conditions
1	-	b
2	a,b	c
3	c	d,a
4	d	-



Example for the analysis of properties:

An automatic soda selling machine accepts 50 c and $1 coins and sells 2 types of products: SODA A, that costs $1.50 and SODA B, that costs $2.00.

Assume that the money return operation is omitted.

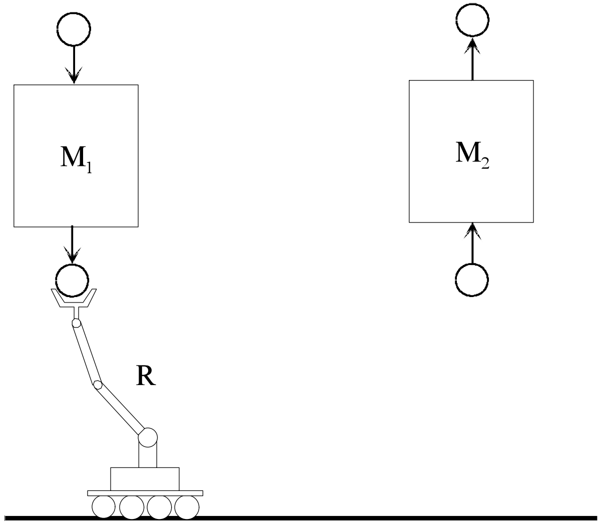


p_1: machine with \$0.00;
t_1: coin of 50 c introduced;
t_8: SODA B sold.

Example for the analysis of properties:

Manufacturing cell with robotic parts handling

Manufacturing system composed by 2 machines (M_1 and M_2) and a robotic manipulator (R). This takes the finished parts from machine M_1 and transports them to M_2.



No buffers available on the machines. If R arrives near M_1 and the machine is busy, the part is rejected.

If R arrives near M_2 and the machine is busy, the manipulator must wait.

Machinning time: M_1=0.5s; M_2=1.5s; $R_{M1 \circledR M2}$=0.2s; $R_{M2 \circledR M1}$=0.1s;

Top 10 Challenges in Logic Control for Manufacturing Systems

By Dawn Tilbury from University of Michigan

10. Distributed Control (General management of distributed control applications, Open/distributed control -- ethernet-based control)

9. Theory (No well-developed and accepted theory of discrete event control, in contrast to continuous control)

8. Languages (None of the programming languages do what we need but nobody wants a new programming language)

7. Control logic synthesis (automatically)

6. Standards (Machine-control standards -- every machine is different, Validated standards, Standardizing different types of control logic programming language)

5. Verification (Standards for validation, Simulation and verification of controllers)

4. Software (Software re-usability -- cut and paste, Sophisticated software for logic control, User-unfriendly software)

3. Theory/Practice Gap (Bridging the gap between industry and academia, Gap between commercial software and academic research)

2. Education (Educating students for various PLCs, Education and keeping current with evolution of new control technologies, Education of engineers in logic control, Lack of curriculum in discrete-event systems)

And the number one challenge in logic control for manufacturing systems is...

1. Diagnostics (Integrating diagnostic tools in logic control, Standardized methodologies for design, development, and implementation of diagnostics)

Complexity and Decidibility

- A problem is *undecidable* if it is proven that no algorithm to solve it exists.

An example of a undecidable problem is the stop of a Turing machine (MT): "Will the TM stops for the code n after using the number m?".

- For *decidable* problems, the complexiy of the solutions have to be taken into account, that is, the computational cost in terms of memory and time.

Basic example: multiplication of number in the arabic and latin civilizations...

Reducibility

When to solve a given problem it is possible to reduce it to other problems with known solution

Theorem: Assume that the problem *A* is reducable to problem B:

Then an instance of A can be transformed in an instance of B:

- If *B* is decidable then *A* is decidable.
- If *A* is undecidable then *B* is undecidable.

Reducibility

Equality Problem: Given two marked Petri nets

$C_1 = (P_1, T_1, I_1, O_1)$ and $C_2 = (P_2, T_2, I_2, O_2)$, with markings

m_1 e m_2, respectively, is $R(C_1, \mu_1) = R(C_2, \mu_2)$?

Subset Problem: Given two marked Petri nets

$C_1 = (P_1, T_1, I_1, O_1)$ and $C_2 = (P_2, T_2, I_2, O_2)$, with markings

m_1 e m_2, respectively, is $R(C_1, \mu_1) \subseteq R(C_2, \mu 2)$?

The equality problem is reducable to the subset problem

(Sugg: proove that each set is a subsets of the other)

Decidibility

If a proble in undecidable does it means that it is not solvable?
NO, it means that it was not yet solved!

Classical example: (Fermat Last Theorem)
$x^n + y^n = z^n$ has solution for n>2 and nontrivial integers x, y e z?
Now it is known that the problem is impossible. The problem remained undecidable for more than 2 centuries (solution proven in 1998).

The MT problem is undecidable.

If it were decidable, for instance the Format last theorem would have been proven long time ago, i.e. there would be an algorithm (MT with code n) that computing all combinations of x,y,z and n>2 (number m) to find a solution verifying $x^n + y^n = z^n$.

Reachability Problems

(Given a Petri net C=(P,T,I,O) with initial marking m)

Reachability Problem: For the marking μ’, is $\mu^{'} \in R(C,\mu)$?

Sub-marking Reachability Problem:

Given the marking μ’ and a subset $P^{'} \subseteq P$, exist $\mu^{''} \in R(C,\mu)$ such that $\mu^{''}(p_i) = \mu^{'} \; \forall p_i \in P^{'}$?

Zero Reachability Problem:

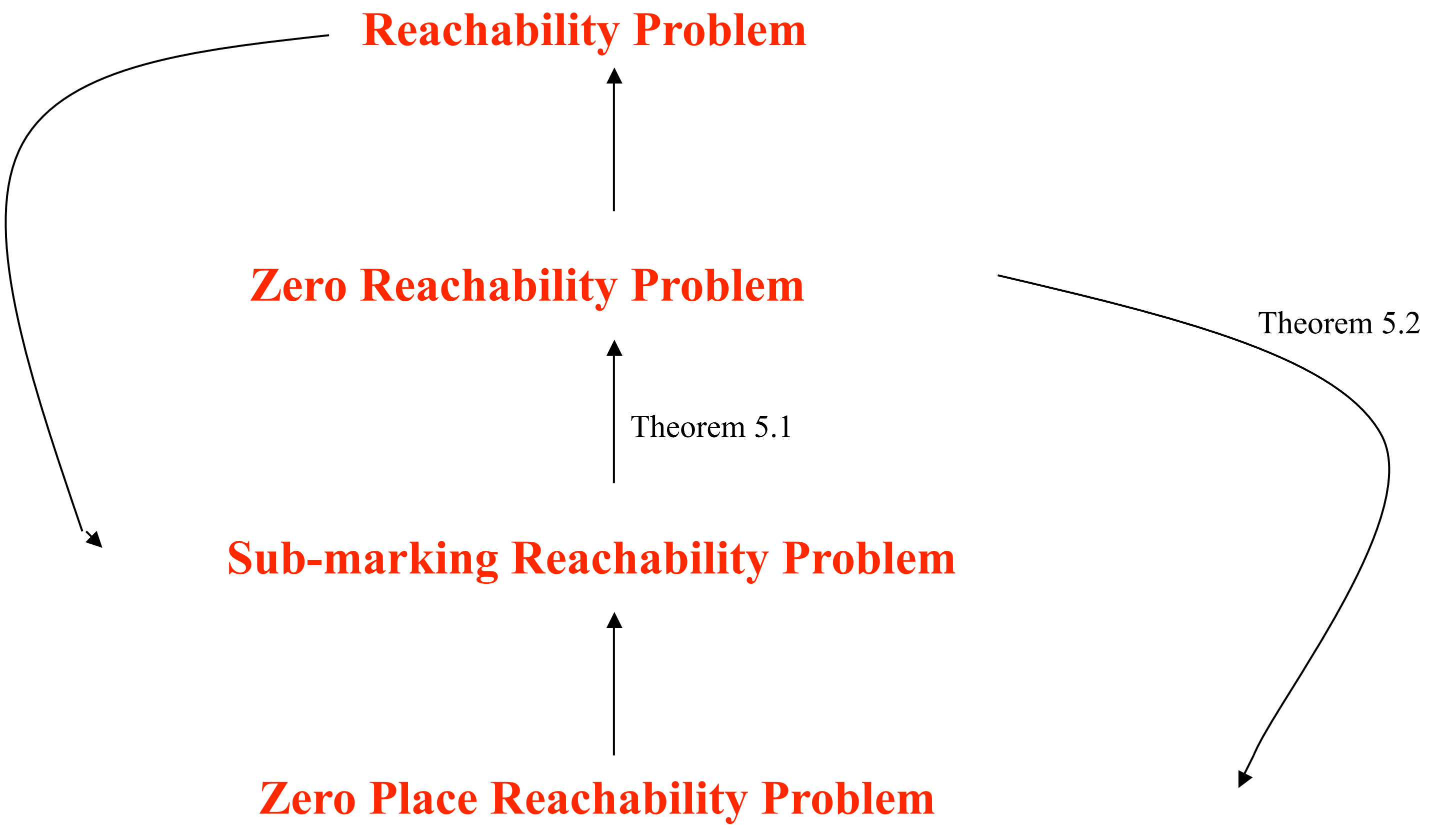
Given the marking μ’=(0 0 … 0), is $\mu^{'} \in R(C,\mu)$?

Zero Place Reachability Problem:

Given the place $p_i \in P$, is $\mu^{'} \in R(C,\mu)$ with $\mu^{'}(p_i) = 0$?

Reachability Problems

A → B : A reducable to B



Reachability Problem

Zero Reachability Problem

Theorem 5.2

Theorem 5.1

Sub-marking Reachability Problem

Zero Place Reachability Problem

Reachability Problems

Theorem 5.3: The following reachability are equivalent:

- **Reachability Problem**;
- **Zero Reachability Problem**;
- **Sub-marking Reachability Problem**;
- **Zero Place Reachability Problem**.

Liveness and Reachability

(Given a Petri net C=(P,T,I,O) with initial marking m)

Liveness Problem

Are all transitions t_j of T live?

Transition Liveness Problem

For the transition t_j of T, is t_j live?

The liveness problem is reducable to the transition liveness problem. To solve the first it remains only to solve the second for the *m* Petri net transitions (#T = *m*).

Liveness and Reachability

(Given a Petri net C=(P,T,I,O) with initial marking m)

Theorem 5.5: The problem of reachability is reducable to the liveness problem.

Theorem 5.6: The problem of liveness is reducable to the reachability problem.

Theorem 5.7: The following problems are equivalent:

- Reachability problem
- Liveness problem

Decidibility results

Theorem 5.10: The sub-marking reachability problem is reducable to the reachable subsets of a Petri net.

Theorem 5.11: The following problem is undecidable:

- Subset problem for reachable sets of a Petri net

They are all reducable to the famous Hilbert's 10th problem:

The solution of the Diophantine equation of n variables, with integer coefficients $P(x_1, x_2, \ldots, x_n)=0$ is undecidable.

(proof by Matijasevic that it is undecidable in the late 1970s).

Industrial Automation

(Automação de Processos Industriais)

DES and Industrial Automation

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

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

Syllabus:

Chap. 7 – Analysis of Discrete Event Systems [2 weeks]

…

Chap. 8 - SEDs and Industrial Automation [1 week]

GRAFCET / Petri Nets Relation

Model modification

Tools adaptation

Analysis of industrial automation solutions by analogy with Discrete Event Systems

…

Chap. 9 – Supervision of DESs [1 week]

Some pointers to Sistemas de Eventos Discretos

History: http://prosys.changwon.ac.kr/docs/petrinet/1.htm

Tutorial: http://www.eit.uni-kl.de/litz/ENGLISH/members/frey/VnVSurvey.htm
http://vita.bu.edu/cgc/MIDEDS/
http://www.daimi.au.dk/PetriNets/

Analysers, and Simulators: http://www.ppgia.pucpr.br/~maziero/petri/arp.html (in Portuguese)
http://wiki.daimi.au.dk:8000/cpntools/cpntools.wiki
http://www.informatik.hu-berlin.de/top/pnk/download.html

Bibliography: * Petri Nets and GRAFCET: Tools for Modelling Discrete Event Systems
R. DAVID, H. ALLA, New York : PRENTICE HALL Editions, 1992

Given a Discrete Event System how to implement it?

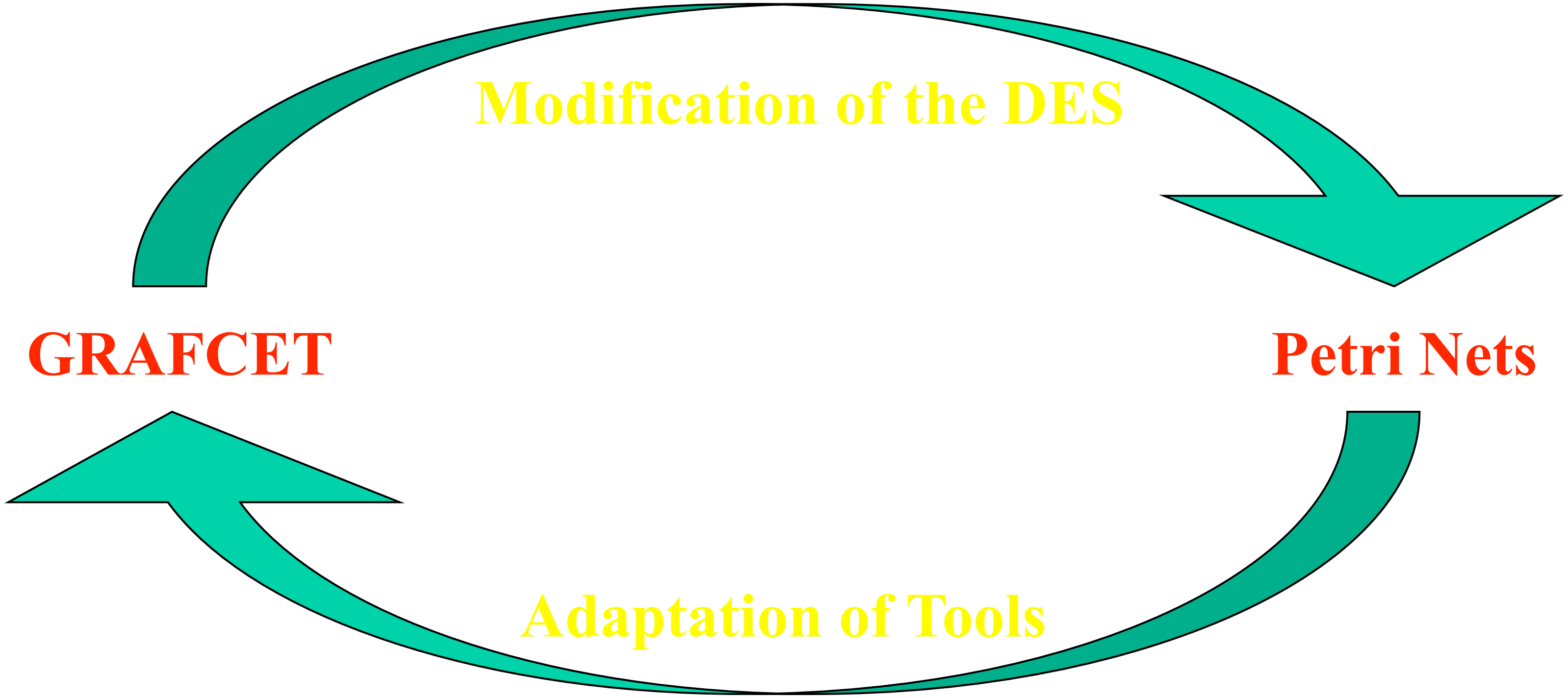
1. Use a GRAFCET
   a) Less modelization hability
   b) Implementation in PLCs straightforward
   c) No analysis (or very scarse) methods available

2. Use a Petri Net
   a) More modelization capacity
   b) No direct implementation in PLCs (therefore indirec
   Or special software solutions required)
   c) Classical analysis methods available

(3. Use an Automata)

Implementation of DES using GRAFCET

ANALYSIS



Modification of the DES

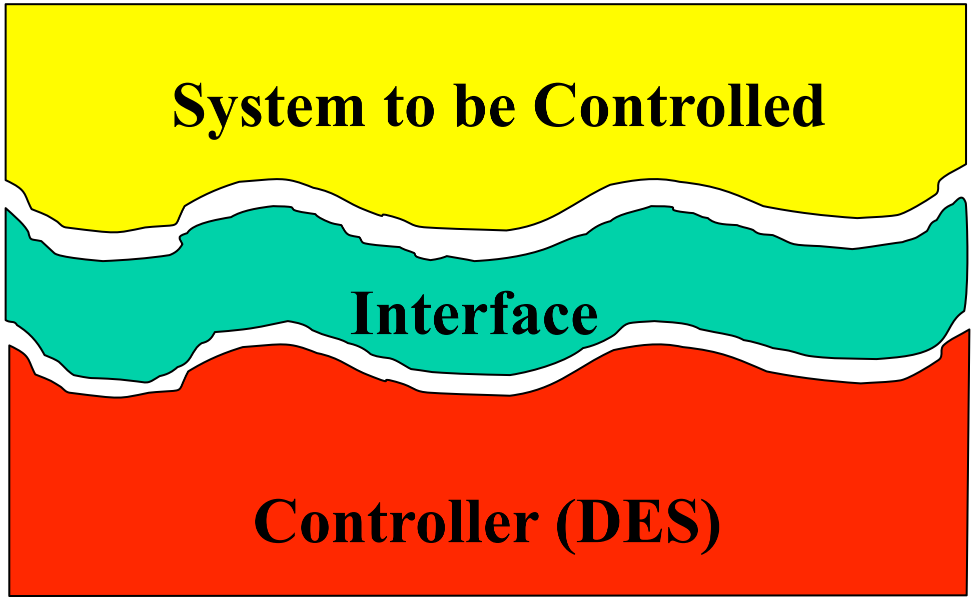
GRAFCET

Petri Nets

Adaptation of Tools

DES Implementation

Models of the DES and of the Controlled system required



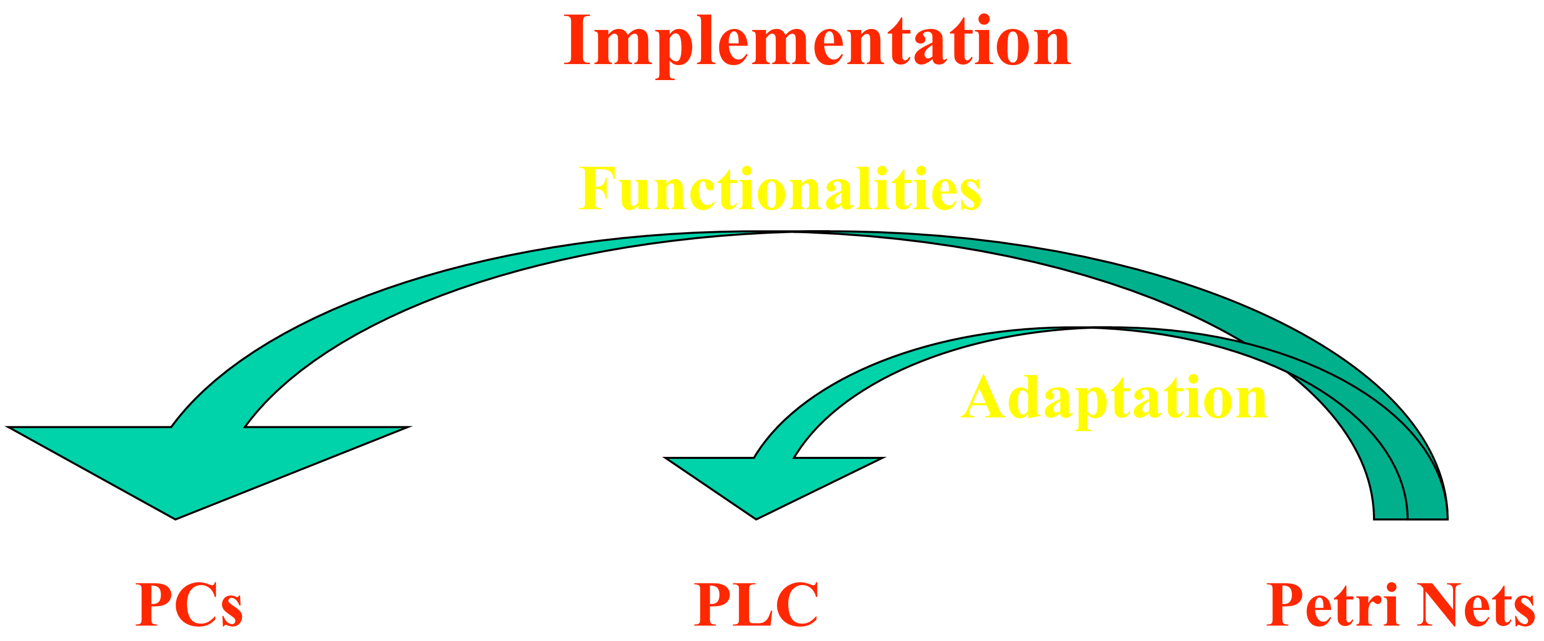
It is required

To design models of the

System to be controlled and of the

Interface to be used...

Implementation of DES using Petri Nets



Both solutions are valid.
Out of the scope of this course.

Analysis of solutions

GRAFCET and Petri Nets

Similarites to exploit:

a) Places and steps are similar

b) Transistions compose both tools

c) Places can be used to implement counters and binary variables

d) Logic functions can be rewritten resorting to the firing of transitions

Analysis of solutions

GRAFCET and Petri Nets

Differences to be taken into account:

a) Firing rules (mutual exclusion)

b) Conflits

c) Binary activation of stages

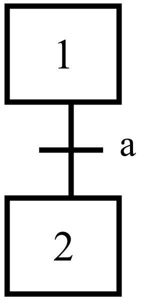
d) Interface with the system to be controlled

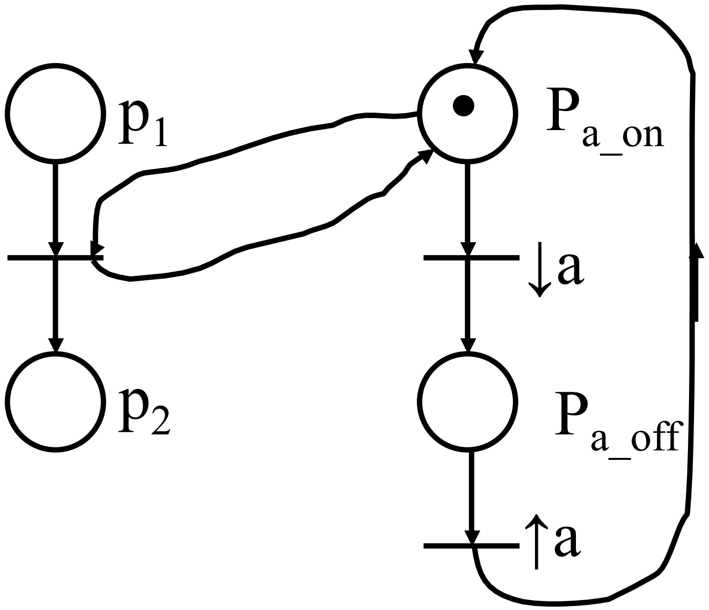
e) Activation functions

Analysis of solutions

GRAFCET → Petri Nets

Representation of variables active on level

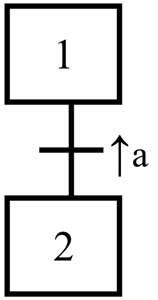


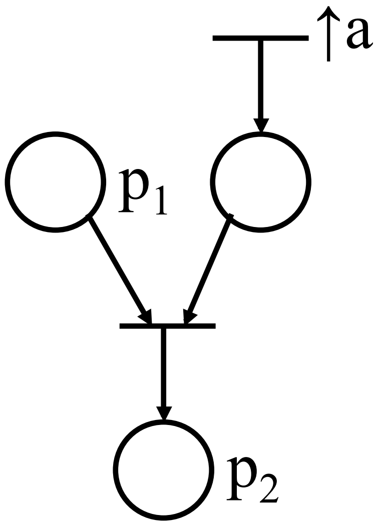


Analysis of solutions

GRAFCET → Petri Nets

Representation of variables active at edge





Note on the memory effects.

Analysis of solutions

Petri Nets → GRAFCET

Adaptation of Tools:

Reachability Tree

⇩

Reachability Graph

**Method of the Matrix Equations
to describe the state evolution**

Petri Nets → GRAFCET

Reachability Graph

To build a graph with the reachable makings.
Composed by two types of nodes:

- terminal

- interior

The duplicated nodes are not represented.
They become connected to the respective copies.

~~The symbol infinity (w) is introduced,~~
~~to obtain finite trees, when a marking covers other(s).~~

Petri Nets → GRAFCET

Reachability Graph

Theorem - If a reachability graph has terminal nodes then the corresponding GRAFCET has deadlocks.

This method will be used to study the properties introduced in Chapter 6.

Petri Nets ⟶ GRAFCET

Reachable Set

Given the GRAFCET G=(S, T, I, O, μ_0) with initial marking μ_0 , the set of all markings that are reachable is the reachable set $\mu' \in R(C,\mu)$.

Remark: IT IS NOT INFINITE!

Given a GRAFCET with m steps it has 2^m nodes at most.

Petri Nets → GRAFCET

Boundness and Limitation

The GRAFCET G=(S, T, I, O, μ_0) is always secure!

The same does not occur with some auxiliary elements of the GRAFCET, e.g., counters and buffers.

For those elements the analysis methods studied for Petri Nets can be used directly.

Petri Nets ⟶ GRAFCET

Conservation

A GRAFCET G=(S, T, I, O, μ_0) is **stricly conservative** if for all m’ ∈ R(C,μ)

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i).$$

A GRAFCET G=(S, T, I, O, μ_0) is **conservative** if there exist a weight vector w, without null elements, for all μ’ ∈ R(C,μ) such that it is constant the quantity

$$\sum_{p_i \in P} w(p_i)\,\mu(p_i).$$

Petri Nets → GRAFCET

Liveness of transições: The transition t_j is live of

Level 0 - it can never be fired.

Level 1 - if it is potentially firable, e.g. if there exist m’ $\in R(C, \mu)$ **such that** t_j **is enabled in** μ’.

Level 2 - if, for each positive *n*, there exist a sequence of firings where occurs *n* firings of t_j **.**

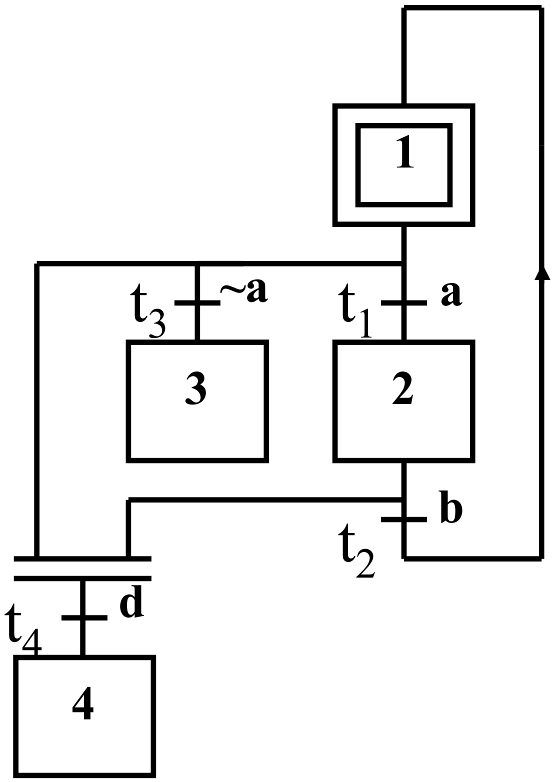
Level 3 - if there exist a sequence of firings where an infinite number of firings of t_j **occurs .**

Level 4 - if for each μ’ $\in R(C, \mu)$ **there exist a sequence** s **that enables the firing of** t_j **.**

Petri Nets → GRAFCET

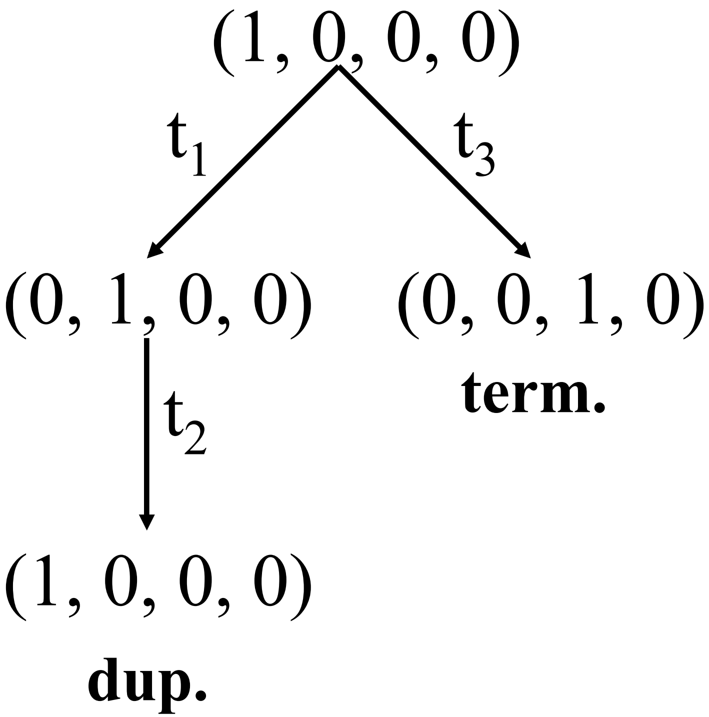
Example of GRAFCET

- t_4 é de nível 0.
- t_1 é de nível 3.
- t_2 é de nível 3.
- t_3 é de nível 1.



Petri Nets ⟶ GRAFCET

Example of GRAFCET



(1, 0, 0, 0)

t_1

t_3

(0, 1, 0, 0)

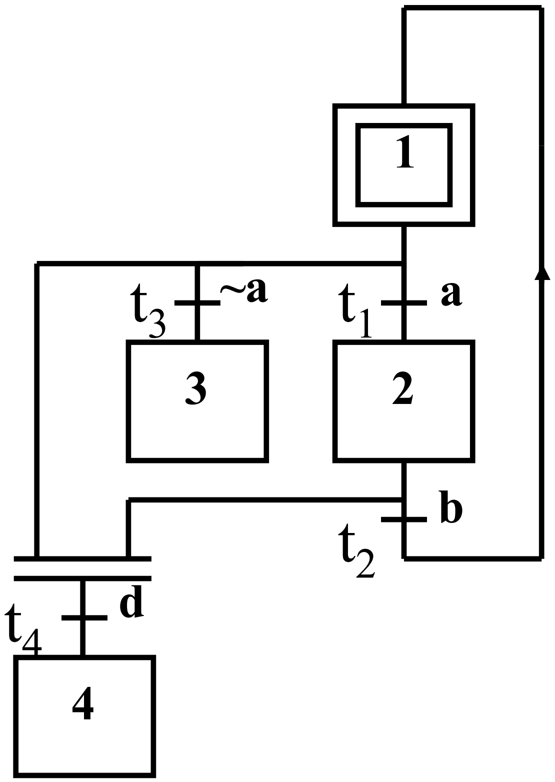
(0, 0, 1, 0)

term.

t_2

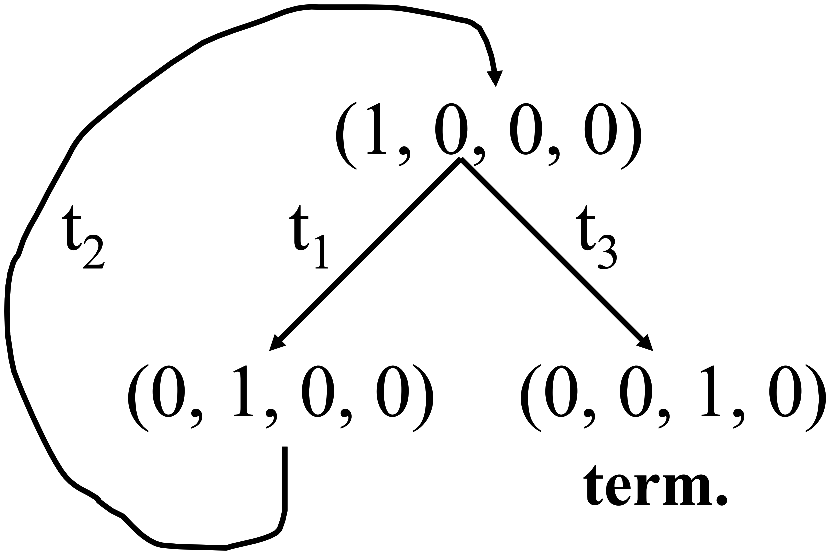
(1, 0, 0, 0)

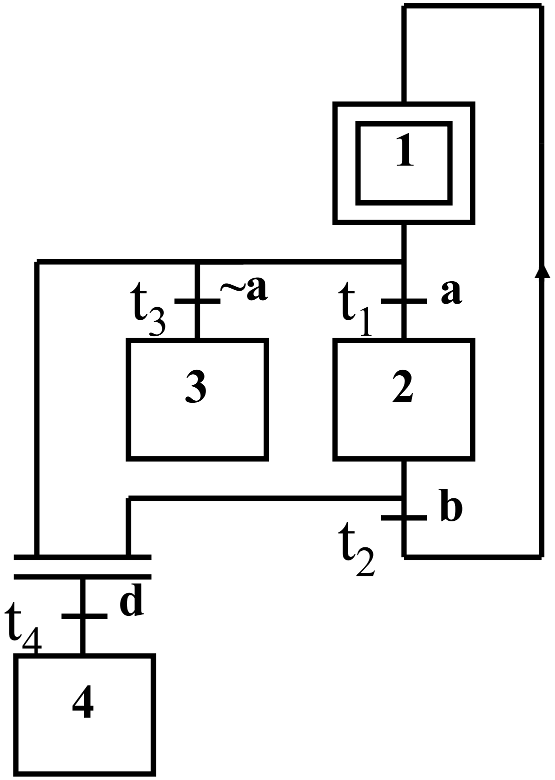
dup.



Petri Nets ⟶ GRAFCET

Example of GRAFCET





Strictly conservative.

Petri Nets → GRAFCET

Metoth of Matrix Equation (for the state evolution)

The evolution of a GRAFCET can be written in compat form as:

$$\mu' = \mu + Dq$$

where:

μ’	- desired marking (vector column vector)
μ	- initial marking
q	- column vector of the transition firings
D	- incidence matrix. Accounts for the token evolution as a consequence of transitions firing.

Petri Nets → GRAFCET

Problems that can be addressed resorting to the Method of Matrix Equations

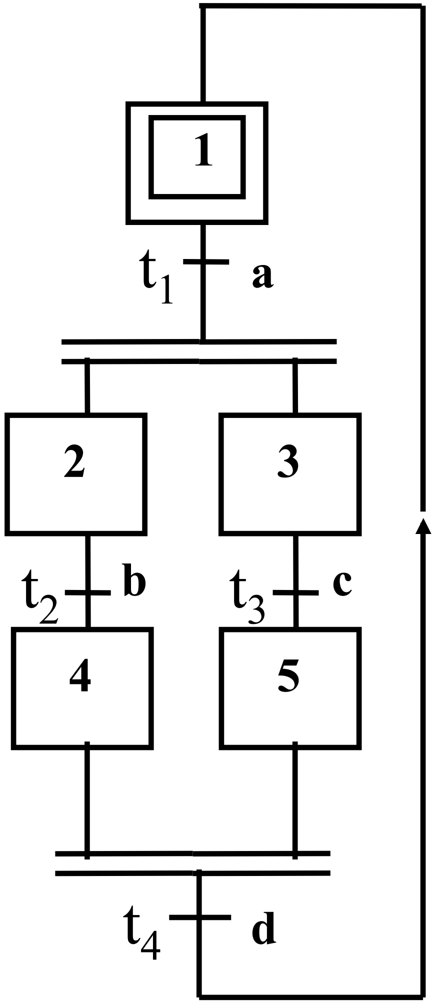
- Reachability (sufficient condition)

Theorem – if the problem of finding the vector of firings, for a GRAFCET without conflicts, from the state μ to the state μ' has no solution using the Method of Matrix Equations, then the problem of reachability of m' is impossible.

- Conservation – the conservation vector can be computed automaticaly.

- Temporal invariance – cycles of operation can be found.

Example of GRAFCET



$$\mu' = \mu + Dq$$

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Conservation $x^T D = 0$

$$\begin{cases} -x_1 + x_2 + x_3 = 0 \\ -x_2 + x_4 = 0 \\ -x_3 + x_5 = 0 \\ x_1 - x_4 - x_5 = 0 \end{cases}$$

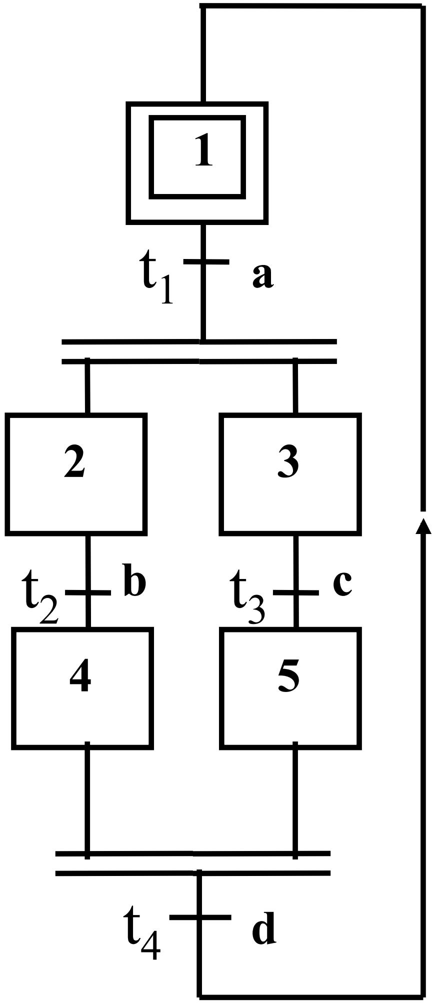
$$x_1 = x_3 + x_4$$
$$x_1 = x_2 + x_5$$
$$x_2 + x_3 = x_4 + x_5$$

Solution:
Undetermined
set of equations

$$x = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Example of GRAFCET

$$\mu' = \mu + Dq$$



$$Dq = 0$$

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad q = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \end{bmatrix}$$

Temporal invariance

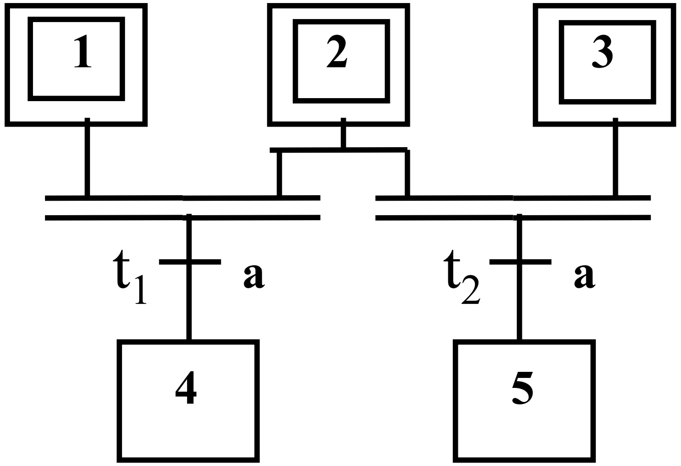
Solution:
Set of equation
with one solution

$$\begin{cases} -\sigma_1 + \sigma_4 = 0 \\ \sigma_1 - \sigma_2 = 0 \\ \sigma_1 - \sigma_3 = 0 \\ \sigma_2 - \sigma_4 = 0 \\ \sigma_3 - \sigma_4 = 0 \end{cases}$$

$$\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 1.$$

Example of GRAFCET

$$\mu' = \mu + Dq$$



$$\mu' = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \mu = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad D = \begin{bmatrix} -1 & 0 \\ -1 & -1 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad q = \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix}$$

$$\begin{cases} 0 = 1 - \sigma_1 \\ 0 = 1 - \sigma_1 - \sigma_2 \\ 0 = 1 - \sigma_2 \\ 1 = \sigma_1 \\ 1 = \sigma_2 \end{cases}$$

Set of Equations implossible
Therefore marking not reachable.
WRONG!

The method fails if it exist conflicts!

Industrial Automation

(Automação de Processos Industriais)

Supervised Control of Discrete Event Systems

http://www.isr.ist.utl.pt/~pjcro/courses/api0910/api0910.html

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

Syllabus:

...

Chap. 8 - SEDs and Industrial Automation [2 weeks]

Chap. 9 – Supervised Control of SEDs [1 semana]

* SCADA

* Methodologies for the Synthesis of Supervision Controllers

* Failure detection

Some jokes available in http://members.iinet.net.au/~ianw/cartoon.html

The End.

Some pointers on Supervised Control of DES

History:

The SCADA Web, http://members.iinet.net.au/~ianw/

Monitoring and Control of Discrete Event Systems
Stéphane Lafortune,
http://www.ece.northwestern.edu/~ahaddad/ifac96/introductory_workshops.html

Tutorial:

http://vita.bu.edu/cgc/MIDEDS/
http://www.daimi.au.dk/PetriNets/

Analysers,
and
Simulators:

http://www.nd.edu/~isis/techreports/isis-2002-003.pdf (Users Manual)
http://www.nd.edu/~isis/techreports/spnbox/ (Software)

Bibliography:

* Livros de SCADA http://www.sss-mag.com/scada.html
* Moody J. e Antsaklis P., “Supervisory Control of Discrete Event Systems using Petri Nets,” Kluwer Academic Publishers, 1998.
* Cassandras, Christos G., "Discrete Event Systems - Modeling and Performance Analysis," Aksen Associates, 1993.
* Yamalidou K., Moody J., Lemmon M. and Antsaklis P.
Feedback Control of Petri Nets Based on Place Invariants
http://www.nd.edu/~lemmon/isis-94-002.pdf

Supervision of DES

Supervisory

Control

And

Data

Acquisition

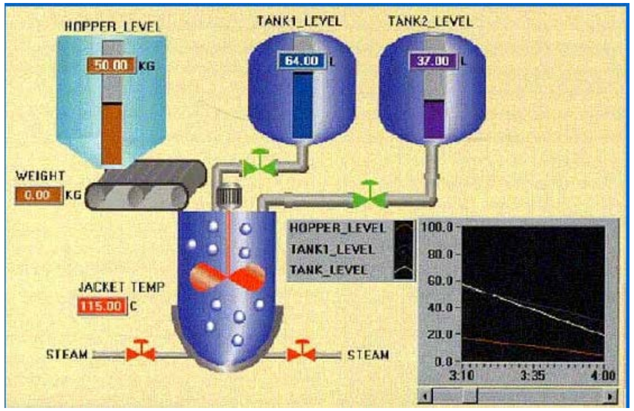
Supervision of DES

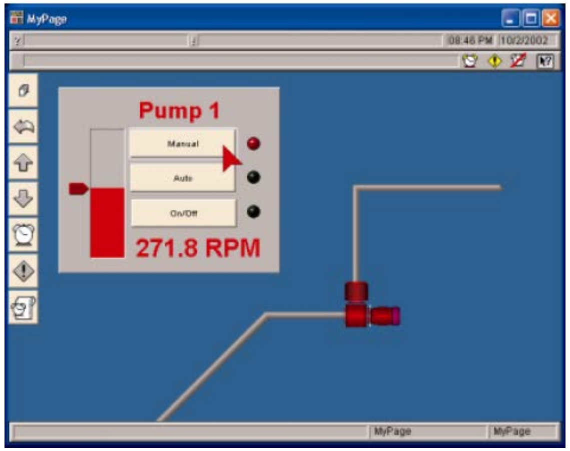
SCADA topics

- Remote monitoring of the state of automation systems
- Logging capacity (resorting to specialized Databases)
- Able to access to *historical* information (plots along time, with selectable periodicity)
- Advanced tools to design Human-Machine interfaces
- Faillure Detection and Isolation capacity (*treshold* and/or logocal functions) on supervised quantities
- Access control

Supervision of DES

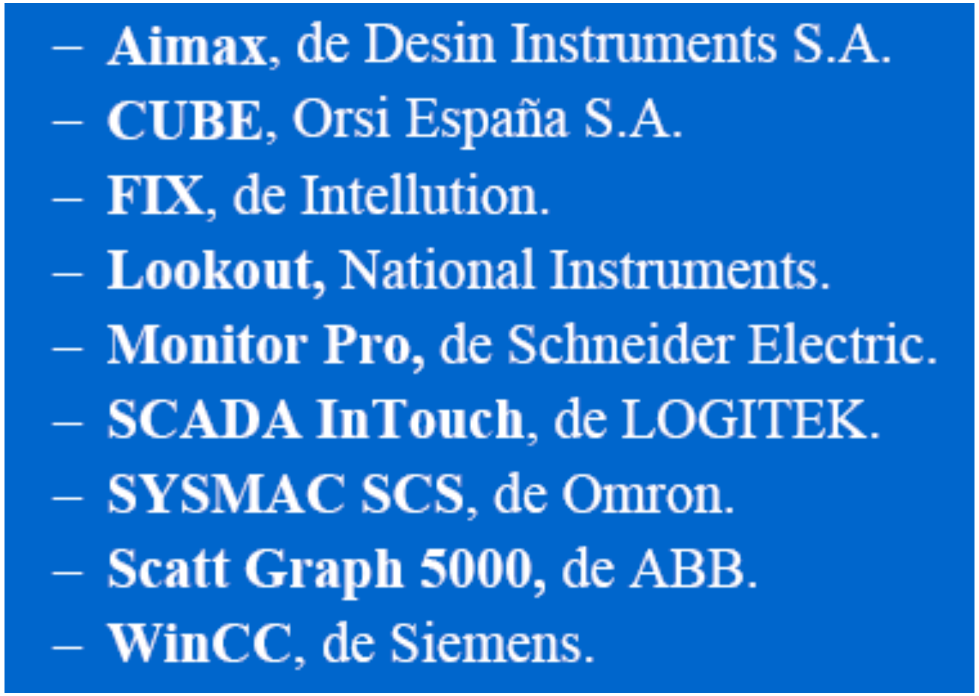
Examples of SCADA





Supervision of DES

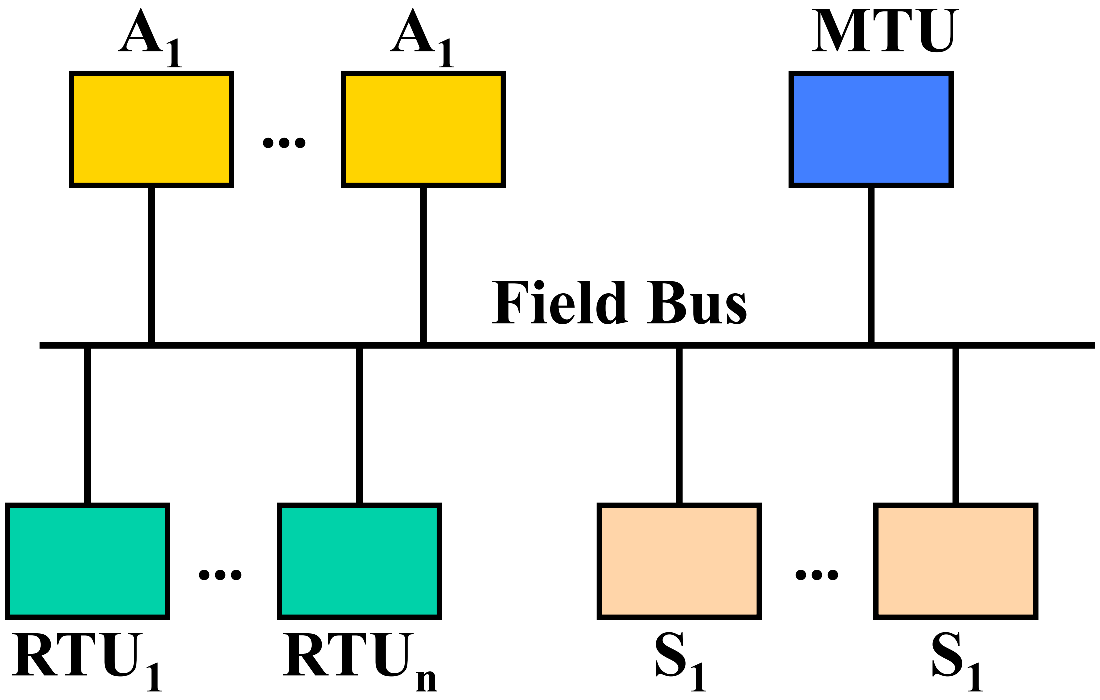
Examples of software packages including SCADA solutions



- **Aimax**, de Desin Instruments S.A.
- **CUBE**, Orsi España S.A.
- **FIX**, de Intellution.
- **Lookout,** National Instruments.
- **Monitor Pro,** de Schneider Electric.
- **SCADA InTouch**, de LOGITEK.
- **SYSMAC SCS**, de Omron.
- **Scatt Graph 5000,** de ABB.
- **WinCC**, de Siemens.

Supervision of DES

Hardware Support Achitecture of SCADA



Legend:

MTU - Main Terminal Unit

RTU - Remote Term. Unit

S – Sensor

A - Actuator

Supervision of DES

And

Now

Something

Completly

Different

Supervision of DES

Objectives of the Supervised Control

- Supervise and bound the work of the supervised DES
- Reinforce that some propeties are verified
- Assure that some states are not reached
- Performance criteria are verified
- Prevent the deadlock od DES
- Constrain on the use of ressources (e.g. mutual exclusion)

Supervision of DES

Some history on Supervised Control

- Methods for finite automata [Ramadge et *al.*], 1989
    - some are based on brute-force search (!)
    - or may require simulation (!)

- Formal verification of *software* in Computer Science (since the 60s) and on *hardware* (90, ...)

- Supervisory Control Method of Petri Nets, method based on *monitors* [Giua et *al.*], 1992.

- Supervisory Control of Petri Nets based on Place Invariants [Moody, Antsaklis et *al.*], 1994 (shares some similitude with the previous one, but deduced independently!...).

Supervision of DES

Advantagens of the Supervisory Control of Petri Nets

- Mathematical representation is clear (and easy)
- Resorts only to linear algebra (matrices)
- More compact then automata
- Straithforward the representation of infinity state spaces
- Intuitive graphical representation available

The representation of the controller as a Petri Net leads to simplified Analysis and Synthesis tasks

Supervision of DES

Method of the Place Invariants [ISIS docs]:

What type of relations can be represented in the method of Place Invariants?

• Sets of linear constraints in the state space

• Representation of convex regions (there are extentions for non-convex regions) (?...)

• Constraints to guarantee liveness and to avoid deadlocks (that can be expressed, in general, as linear constraints)

• Constraints on the events and timmings (bis)

Supervision of DES

Advantages of the Method of the Place Invariants [ISIS docs]:

Other characteristics that can impact on the solutions?

- Existence and uniqueness
- Optimality of the solutions (e.g. see maximal permissivity next)
- Existence of transition non-controllable and/or not observable (remind definitions for time-driven systems)

In general the solutions can be found solving:

Linear Programming Problems, with Linear Constraints

Methods of Analysis/Synthesis

Method of the Matrix Equations (just to remind)

The dynamics of the Petri net state can be written in compact form as:

$$\mu(k+1) = \mu(k) + Dq(k)$$

where:

$\mu(k+1)$	- marking to be reached
$\mu(k)$	- initial marking
$q(k)$	- firing vector (transitions)
D	- incidence matrix. Accounts the balance of tokens, giving the transitions fired.

Methods of Analysis/Synthesis

How to build the Incidence Matrix?

For a Petri net with *n* places and *m* transitions

$$\mu \in N_0^{\ n}$$

$$q \in N_0^{\ m}$$

$$D = D^+ - D^- \qquad \in \mathrm{Z}^{n\times m}$$

The enabling firing rule is $\mu \geq D^- q.$

Can also be writen in compact form as the inequality

$$\mu + Dq \geq 0,$$

interpreted element by element.

Methods of Synthesis

Some notation for the method

- The supervised system is modelled as a Petri net with n places and m transitions, and incidence matrix

$$\boxed{D_P \in \mathrm{Z}^{n \times m}.}$$

- The supervisor is modelled as a Petri net with n_C places and m transitions, and incidence matrix

$$\boxed{D_C \in \mathrm{Z}^{n_C \times m}.}$$

- The resulting total system has an incidence matrix

$$\boxed{D \in \mathrm{Z}^{(n+n_C) \times m}.}$$

Methods of Synthesis

Theorem: (1)
Synthesis of Controllers based on Place Invariants

Given the set of linear state constraints that the supervised system must follow, writte as

$$\boxed{L\mu_P \le b, \quad \mu_P \in N_0{}^n, \quad L \in Z^{n_C \times n} \quad and \quad b \in Z^{n_C}.}$$

If $b - L\mu_{P_0} \ge 0$, then the controller with incedence matrix and initial marking, respectively

$$\boxed{D_C = -LD_P,} \quad \text{and} \quad \boxed{\mu_{C_0} = b - L\mu_{P_0},}$$

makes the constraints be verified for all markings obtained from the initial marking.

Methods of Synthesis

Theorem:

Proof outline:

The constraint $L\mu_P \leq b$ can be written as
$L\mu_P + \mu_C = b,$ using the slack variables $\mu_C.$
They represent the marking of the n_C places of the controller.
To have a place invariant, the relation $x^T D = 0$ must be verified and in particular, given the previous constraint:

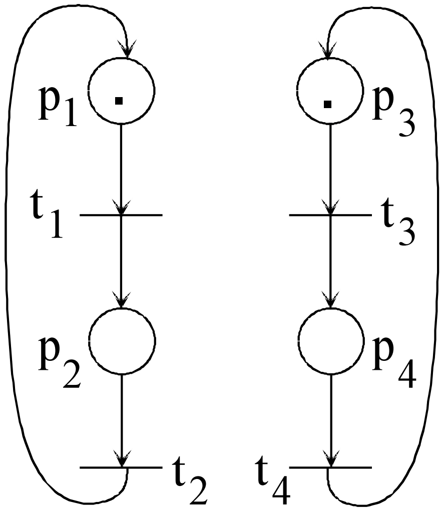
$$x^T D = \begin{bmatrix} L & I \end{bmatrix} \begin{bmatrix} D_P \\ D_C \end{bmatrix} = 0, \text{ resulting } \quad \boxed{D_C = -LD_P.}$$

From $L\mu_{P_0} + \mu_{C_0} = b,$ follows that $\boxed{\mu_{C_0} = b - L\mu_{P_0}.}$ ▮

Methods of Synthesis

Example of controller synthesis

Mutual Exclusion



Linear constraint: $\mu_2 + \mu_4 \leq 1$

That can be written as:

$$L\mu_P \leq b \qquad \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} \leq 1.$$

Incidence Matrix $D_P = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$ and initial marking $\mu_{P_0} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$

Methods of Synthesis

Example of controller synthesis

Mutual Exclusion

1) Test

$$b - L\mu_{P_0} = 1 - \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 1 \geq 0.$$

OK.

2) Compute

$$D_C = -LD_P = -\begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix},$$

and

$$\mu_{C_0} = b - L\mu_{P_0} = 1.$$

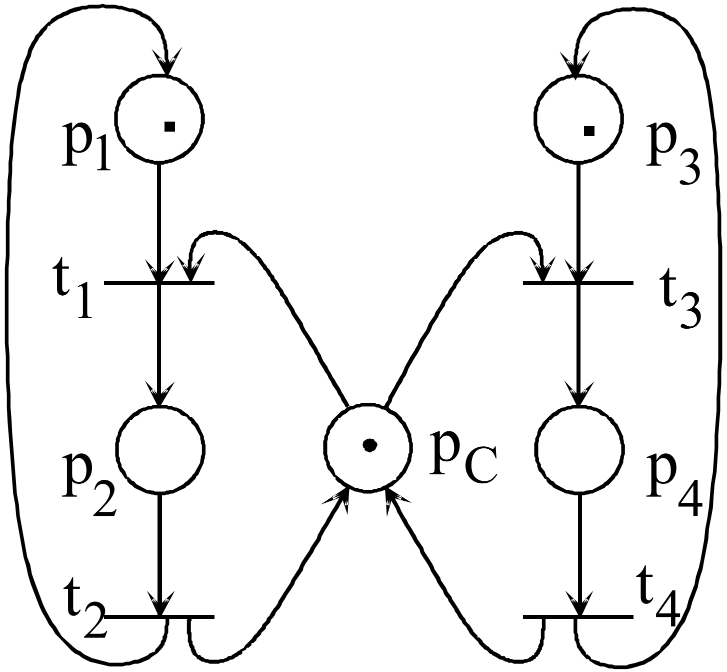
OK.

Methods of Synthesis

Example of controller synthesis

Mutual Exclusion

3) Resulting in



$$D = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

$$\mu_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

OK.
UAU!!!.

Methods of Synthesis

Definition:

Maximal permissivity occurs when all the linear constraints are verified and all legal markings can be reached.

Lemmas:

i) The controllers obtained in (1) have maximal permissivity.

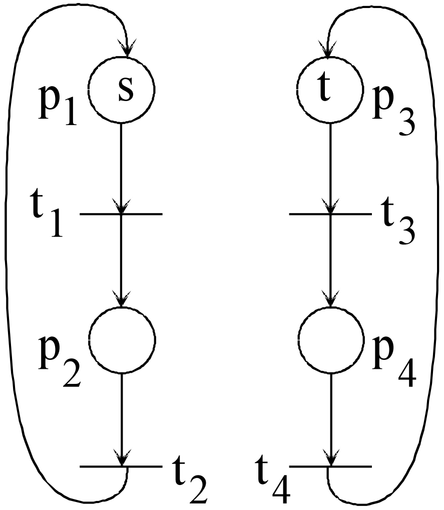
ii) Given the linear constraints used, the place invariants obtained with the controller synthesized with (1) are the same as the invariants associated with the initial system.

Methods of Synthesis

Example of controller synthesis

$$\forall s \in N_0, \forall t \in N_0, \forall n \in N_0$$

Readers / Writers



Linear constraints for *n* books: $\mu_2 + n\mu_4 \le n$

That can be written as:

$$L\mu_P \le b \qquad \begin{bmatrix} 0 & 1 & 0 & n \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} \le n.$$

Incidence Matrix $D_P = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$ and initial marking $\mu_{P_0} = \begin{bmatrix} s \\ 0 \\ t \\ 0 \end{bmatrix}.$

Methods of Synthesis

Example of controller synthesis

Readers / Writers

1) Test

$$b - L\mu_{P_0} = n - \begin{bmatrix} 0 & 1 & 0 & n \end{bmatrix} \begin{bmatrix} s \\ 0 \\ t \\ 0 \end{bmatrix} = n \geq 0.$$

OK.

2) Compute

$$D_C = -LD_P = -\begin{bmatrix} 0 & 1 & 0 & n \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -n & n \end{bmatrix},$$

and

$$\mu_{C_0} = b - L\mu_{P_0} = n.$$

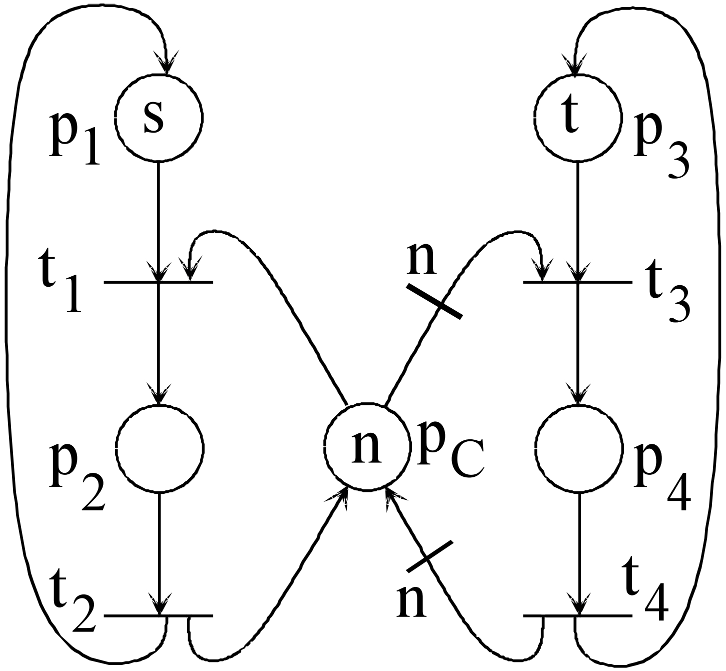
OK.

Methods of Synthesis

Example of controller synthesis

Readers / Writers

3) Resulting in



$$D = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & -n & n \end{bmatrix}$$

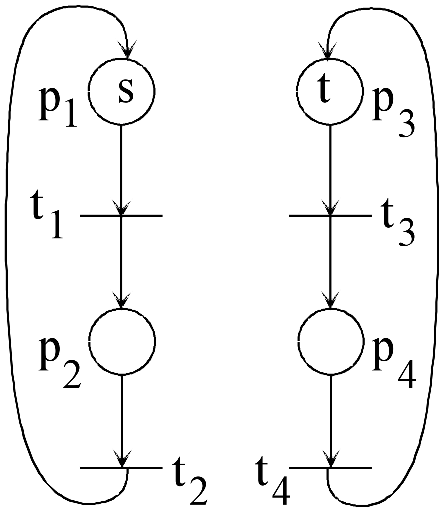
$$\mu_0 = \begin{bmatrix} s \\ 0 \\ t \\ 0 \\ n \end{bmatrix}$$

OK.
UAU!!!.

Methods of Synthesis

Example of controller synthesis

Producer / Consumer



Incidence matrix

$$D_P = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Initial marking

$$\mu_{P_0} = \begin{bmatrix} s \\ 0 \\ t \\ 0 \end{bmatrix}.$$

What is the linear constraint?

Not possible to write it as a linear constraint $L\mu_P \leq b$!

Is it impossible to solve this problem with the proposed method?

Methods of Synthesis

Generalized linear constraint

Let the generalized linear constraint be

$$\boxed{\begin{array}{c} L\mu_P + F q_P + C v_P \leq b, \\ \mu_P \in N_0{}^n, v_P \in N_0{}^m, q_P \in N_0{}^m, \\ L \in Z^{n_C \times n}, F \in Z^{n_C \times m}, C \in Z^{n_C \times m}, e \quad b \in Z^{n_C}\ , \end{array}}$$

where

* μ_P is the marking vector for system P;

* q_P is the firing vector since t_0;

* v_P is the number of transtitions (firing) that can occur, also designated as Parikh vector.

Methods of Synthesis

Function LINENF of SPNBOX

Theorem: Synthesis of Controllers based on Place Invariants, for Generalized Linear Contraints

Given the generalized linear constraint $\boxed{L\mu_P + F q_P + C v_P \leq b,}$

if $b - L\mu_{P_0} \geq 0$, then the controller with incidence matrix and initial marking, respectively

$$\boxed{\begin{aligned} D_C^- &= \max\left(0, LD_P + C, F\right) \\ D_C^+ &= \max\left(0, F - \max\left(0, LD_P + C\right)\right) - \min\left(0, LD_P + C\right), \end{aligned}}$$

$$\boxed{\mu_{C_0} = b - L\mu_{P_0} - C v_{P0},}$$

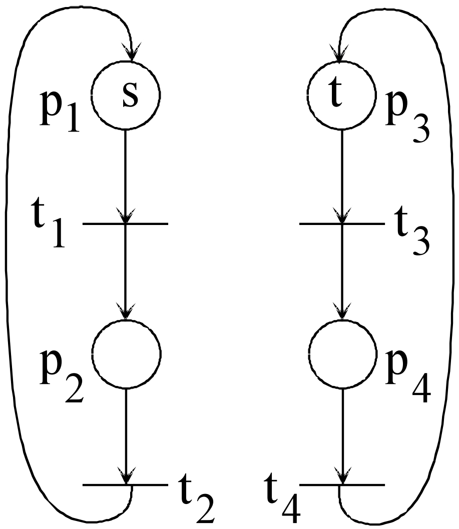
guarantees that constraints are verified for the states resulting from the initial marking.

Methods of Synthesis

Example of controller synthesis

$\forall s \in N_0, \forall t \in N_0, \forall n \in N_0$

Producer / Consumer



Linear constraint: $v_3 \leq v_2$

That can be written as:

$$Cv_P \leq b$$

$$L = 0, F = 0$$

$$\begin{bmatrix} 0 & -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \leq 0.$$

Incidence matrix

$$D_P = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Initial marking

$$\mu_{P_0} = \begin{bmatrix} s \\ 0 \\ t \\ 0 \end{bmatrix}.$$

Methods of Synthesis

Example of controller synthesis

Producer / Consumer

1) Test

$$b - L\mu_{P_0} = 0 - 0 \geq 0.$$

OK.

2) Compute

$$D_C^- = \max\left(0, \begin{bmatrix}0 & -1 & 1 & 0\end{bmatrix}, 0\right) = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix},$$

$$D_C^+ = \max\left(0, -\begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}\right) - \min\left(0, \begin{bmatrix}0 & -1 & 1 & 0\end{bmatrix}\right) =$$

$$= \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix} - \begin{bmatrix}0 & -1 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}$$

e

$$\mu_{C_0} = b - L\mu_{P_0} = 0 - 0 = 0.$$

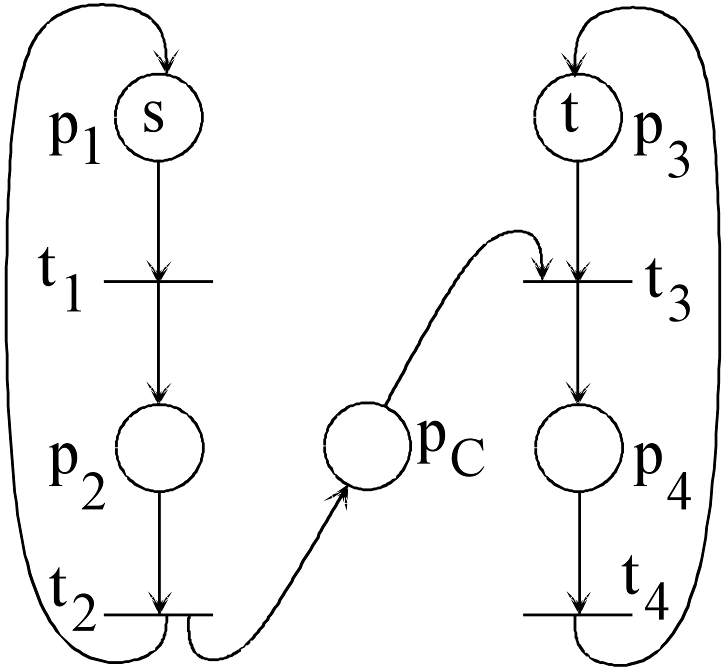
OK.

Methods of Synthesis

Example of controller synthesis

Producer / Consumer

3) Resulting in



$$D = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

$$\mu_0 = \begin{bmatrix} s \\ 0 \\ t \\ 0 \\ 0 \end{bmatrix}$$

OK.

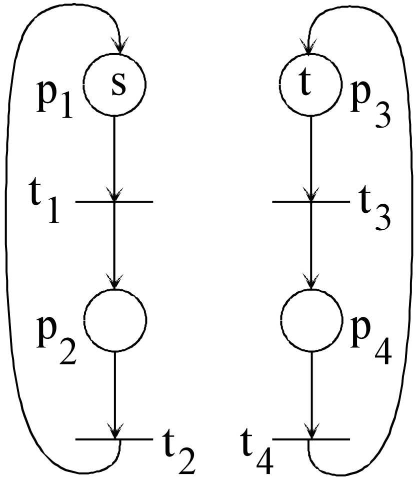
UAU!!!.

Methods of Synthesis

$$\forall s \in N_0, \forall t \in N_0, \forall n \in N_0$$

Example of controller synthesis

Bounded
Producer / Consumer



TWO linear constraints:

$$v_2 - v_3 \leq n$$

$$v_3 - v_2 \leq n$$

That can be written as:

$$Cv_P \leq b \qquad \begin{bmatrix} 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \leq \begin{bmatrix} n \\ n \end{bmatrix}.$$

$$L = 0, F = 0$$

Incidence matrix

$$D_P = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Initial marking

$$\mu_{P_0} = \begin{bmatrix} s \\ 0 \\ t \\ 0 \end{bmatrix}.$$

Methods of Synthesis

Example of controller synthesis

Bounded Producer / Consumer

1) Test

$$b - L\mu_{P_0} = \begin{bmatrix} n \\ n \end{bmatrix} \geq 0.$$

OK.

2) Compute

$$D_C^- = \max\left(0, \begin{bmatrix} 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix}, 0\right) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$D_C^+ = \max\left(0, 0 - \max\left(0, \begin{bmatrix} 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix}\right)\right) - \min\left(0, \begin{bmatrix} 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix}\right) =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0\mathrm{e} \end{bmatrix} - \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

$$\mu_{C_0} = b - L\mu_{P_0} = \begin{bmatrix} n \\ n \end{bmatrix}.$$

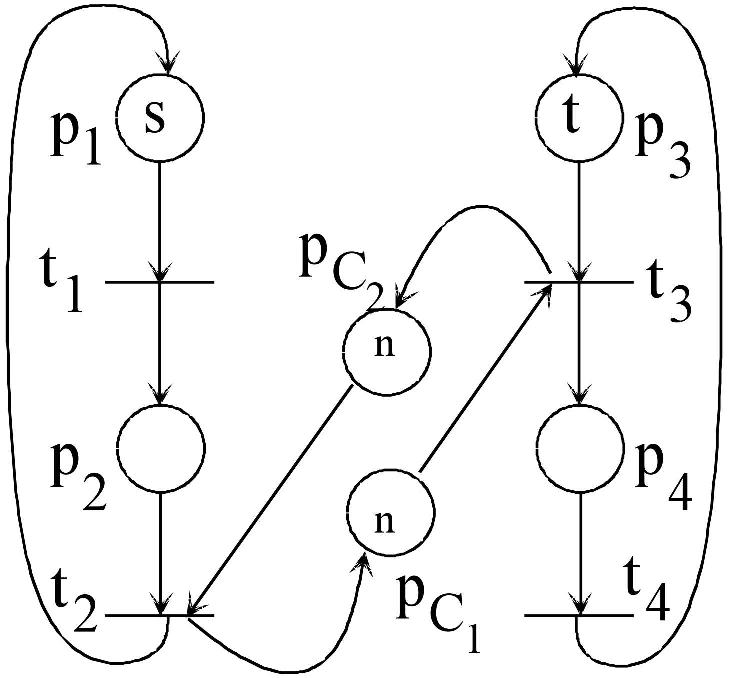
OK.

Methods of Synthesis

Example of controller synthesis

Bounded Producer / Consumer

3) Resulting in



$$D = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix}$$

$$\mu_0 = \begin{bmatrix} s \\ 0 \\ t \\ 0 \\ n \\ n \end{bmatrix}$$

OK.
UAU!!!.

Methods of Synthesis

Definition of Uncontrolable Transition:

A transition is uncontrolable if its firing can be inhibited by an external action (e.g. A supervisory controller).

Definition of Unobservable Transition:

A transition is unobservable if its firing can not be detected or measured (therefore the study of any supervisory controller can not depend from that firing).

Proposition:

A controller can not have arcs that connect to unobservable Transitions, then all unobservable transitions are implicitly uncontrolables.

Methods of Synthesis

Definition: A marking is admissible if

i) $L\mu_P \leq b,$

e

ii)

$$\forall \mu' \in R(C, \mu_{P_0}) \quad verif\ ies \quad L\mu' \leq b.$$

Definition: A Linear Constraint is admissible if

i) $L\mu_{P_0} \leq b,$

and

ii) $\forall \mu' \in R(C, \mu_{P_0}) \quad such\, that \quad L\mu' \leq b,$

is an admissible marking.

Methods of Synthesis

Proposition: Admissibility of a constraint

A linear constraint is admissible iff

- The initial markings satisfy the constraint.
- There exist a controller with maximal permissivity that forces the constraint and does not inhibit any uncontrolable transition.

Corolary: given a system with uncontrolable transitions, $\boxed{l^T D_{uc} \leq 0}$ implies admissibility.

Corolary: given a system with unobservable transitions, $\boxed{l^T D_{uo} = 0}$ implies admissibility.

Methods of Synthesis

Function MRO_ADM da SPNBOX

Lemma: Structure of Constraint transformation

Let $R_1 \in Z^{n_C \times n}$ *such that* $R_1 \mu_P \geq 0,$

$R_2 \in Z^{n_C \times n_C}$ be a matrix with positive elements in the diagonal,

If there exists
$$L^{'} = R_1 + R_2 L$$
$$b^{'} = R_2 (b+1) - 1,$$

such that $L^{'} \mu_P \leq b^{'}$

then it is also verified that $L \mu_P \leq b.$

The End.