Industrial Automation
(Automação de Processos Industriais)

PLC Programming languages
Ladder Diagram

http://users.isr.ist.utl.pt/~jag/courses/api1213/api1213.html

Slides 2010/2011 Prof. Paulo Jorge Oliveira
Rev. 2011-2013 Prof. José Gaspar
Syllabus:

Chap. 2 – Introduction to PLCs [2 weeks]

Chap. 3 – PLC Programming languages [2 weeks]

Chap. 4 - GRAFCET (Sequential Function Chart) [1 week]
PLC Programming languages
(IEC 1131-3 changed to IEC 61131-3)

Ladder Diagram

Structured Text

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

Instruction List

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

Sequential Function Chart
(GRAFCET)
A program is a series of instructions that directs the PLC to execute actions.

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

Types of operands:
# Types of operands:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Examples</th>
<th>Write access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate values</td>
<td>0 or 1 (False or True)</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
| Inputs/outputs           | These bits are the "logic images" of the electrical states of the inputs/outputs. They are stored in the data memory and updated each time the task in which they are configured is polled. Note: The unused input/output bits may not be used as internal bits. | %I23.5  
%Q51.2 | No  
Yes |
| Internal                 | The internal bits are used to store the intermediary states during execution of the program. | %M200 | Yes |
| System                   | The system bits %S0 to %S127 monitor the correct operation of the PLC and the running of the application program. | %S10 | According to i |
| Function blocks          | The function block bits correspond to the outputs of the function blocks or DFB instance. These outputs may be either directly connected or used as an object. | %TM8.Q | No |
| Word extracts            | With the PL7 software it is possible to extract one of the 16 bits of a word object. | %MW10:X5 | According to the type of words |
| Grafcet steps and macro-steps | The Grafcet status bits of the steps, macro-steps and macro-step steps are used to recognize the Grafcet status of step i, of macro-step j or of step i of the macro-step j. | %X21  
%X5.9  | Yes  
Yes |
Ladder diagram

Basic Instructions

**Load**

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

- **Normally closed** contact: contact is active (result is 1) when the control bit is 0.

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

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

![Ladder diagram Ladder diagram](image-url)
### Ladder diagram

#### Basic Instructions

**Load operands**

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

<table>
<thead>
<tr>
<th>Ladder</th>
<th>Instruction list</th>
<th>Structured text</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Ladder Diagram" /></td>
<td>LD</td>
<td>:=</td>
<td>%I,%Q,%M,%S,%BLK,%*:Xk, %Xi, (True and False in instruction list or structured text)</td>
</tr>
<tr>
<td><img src="image" alt="Ladder Diagram" /></td>
<td>LDN</td>
<td>:=NOT</td>
<td>%I,%Q,%M,%S,%BLK,%*:Xk, %Xi, (True and False in instruction list or structured text)</td>
</tr>
<tr>
<td><img src="image" alt="Ladder Diagram" /></td>
<td>LDR</td>
<td>:=RE</td>
<td>%I,%Q,%M</td>
</tr>
<tr>
<td><img src="image" alt="Ladder Diagram" /></td>
<td>LDF</td>
<td>:=FE</td>
<td>%I,%Q,%M</td>
</tr>
</tbody>
</table>
Ladder diagram

Basic Instructions

**Store**  
\[\text{The result of the logic function activates the coil.}\]

\[\text{The inverse result of the logic function activates the coil.}\]

\[\text{The result of the logic function energizes the relay (sets the latch).}\]

\[\text{The result of the logic function de-energizes the relay (resets the latch).}\]

![Ladder diagram](image-url)
## Ladder diagram

### Basic Instructions

#### Store operands

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

<table>
<thead>
<tr>
<th>Permitted operands</th>
<th>Language data</th>
<th>Instruction list</th>
<th>Structured text</th>
<th>Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ST</td>
<td>:=</td>
<td>%I,%Q,%M,%S,%:*Xk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STN</td>
<td>:=NOT</td>
<td>%I,%Q,%M,%S,%:*Xk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
<td>SET</td>
<td>%I,%Q,%M,%S,%:*Xk,%Xi Only in the preliminary processing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>RESET</td>
<td>%I,%Q,%M,%S,%:*Xk,%Xi Only in the preliminary processing.</td>
</tr>
</tbody>
</table>
Ladder diagram

*Allen Bradley notation*

Relays with *latch* and *unlatch*
Ladder diagram

Relay-type instructions

Example:
Ladder diagram

Basic Instructions

**AND**

- AND of the operand with the result of the previous logical operation.

- AND of the operand with the inverted result of the previous logical operation.

- 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

\textit{XOR}

<table>
<thead>
<tr>
<th>Instruction list</th>
<th>Structured text</th>
<th>Description</th>
<th>Timing diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>XOR</td>
<td>OR Exclusive between the operand and the previous instruction's Boolean result.</td>
<td></td>
</tr>
<tr>
<td>%I1.1 %M1</td>
<td></td>
<td></td>
<td><img src="image1" alt="Timing Diagram for XOR" /></td>
</tr>
<tr>
<td>%M1 %I1.1</td>
<td></td>
<td></td>
<td><img src="image2" alt="Timing Diagram for XOR" /></td>
</tr>
<tr>
<td>%M2 %I1.2</td>
<td></td>
<td></td>
<td><img src="image3" alt="Timing Diagram for XOR" /></td>
</tr>
<tr>
<td>%I1.2 %M2</td>
<td></td>
<td></td>
<td><img src="image4" alt="Timing Diagram for XOR" /></td>
</tr>
<tr>
<td>XORN</td>
<td>XOR (NOT...)</td>
<td>OR Exclusive between the operand inverse and the previous instruction's Boolean result.</td>
<td></td>
</tr>
<tr>
<td>%I1.1</td>
<td></td>
<td></td>
<td><img src="image5" alt="Timing Diagram for XORN" /></td>
</tr>
<tr>
<td>%M1</td>
<td></td>
<td></td>
<td><img src="image6" alt="Timing Diagram for XORN" /></td>
</tr>
<tr>
<td>%Q2.3</td>
<td></td>
<td></td>
<td><img src="image7" alt="Timing Diagram for XORN" /></td>
</tr>
<tr>
<td>XORR</td>
<td>XOR (RE...)</td>
<td>OR Exclusive between the operand's rising edge and the previous instruction's Boolean result.</td>
<td></td>
</tr>
<tr>
<td>%I1.3</td>
<td></td>
<td></td>
<td><img src="image8" alt="Timing Diagram for XORR" /></td>
</tr>
<tr>
<td>%I1.4</td>
<td></td>
<td></td>
<td><img src="image9" alt="Timing Diagram for XORR" /></td>
</tr>
<tr>
<td>%Q2.4</td>
<td></td>
<td></td>
<td><img src="image10" alt="Timing Diagram for XORR" /></td>
</tr>
<tr>
<td>XORF</td>
<td>XOR (FE...)</td>
<td>OR Exclusive between the operand's falling edge and the previous instruction's Boolean result.</td>
<td></td>
</tr>
<tr>
<td>%M3</td>
<td></td>
<td></td>
<td><img src="image11" alt="Timing Diagram for XORF" /></td>
</tr>
<tr>
<td>%I1.5</td>
<td></td>
<td></td>
<td><img src="image12" alt="Timing Diagram for XORF" /></td>
</tr>
<tr>
<td>%Q2.5</td>
<td></td>
<td></td>
<td><img src="image13" alt="Timing Diagram for XORF" /></td>
</tr>
</tbody>
</table>
Ladder diagram

Ladder assembling

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

Example:

![Ladder Diagram Example](image-url)
Ladder diagram

Example:

(a) Hard-wired circuit

(b) Programmed circuit

Seal-in circuit.
Ladder diagram

Example:

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

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

Imbricated (nested) contacts and alternative solution

Fig. 5-25
Nested contact program.

Fig. 5-26
Program required to eliminate nested contact.
Contacts in the vertical and alternative solution

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

**Fig. 5-28**
Program with vertical contact

**Fig. 5-29**
Reprogrammed to eliminate vertical contact.
Ladder diagram

Contacts in the vertical and alternative solution

Another example:

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

**Fig. 5-30**
Original circuit.

**Fig. 5-31**
Reprogrammed circuit.

Solves the problem of disallowed right to left scanning (FDBC in fig5.30).
Ladder diagram  Temporized Relays or Timers

Solid-state timing relay  Pneumatic timing relay  Plug-In timing relay
The instantaneous contacts change state as soon as the timer coil is powered. The delayed contacts change state at the end of the time delay.
**Ladder diagram**  
*Temporized Relays or Timers*

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

<table>
<thead>
<tr>
<th>On-delay symbols</th>
<th>Off-delay symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="On-delay symbol" /> or <img src="image" alt="On-delay symbol" /></td>
<td><img src="image" alt="Off-delay symbol" /> or <img src="image" alt="Off-delay symbol" /></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normally open, timed closed contact (NOTC). Contact is open when relay coil is de-energized. When relay is energized, there is a time delay in closing.</td>
<td>Normally closed, timed open contact (NCTO). Contact is closed when relay coil is de-energized. When relay is energized, there is a time delay in opening.</td>
</tr>
</tbody>
</table>

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Off-delay symbol" /> or <img src="image" alt="Off-delay symbol" /></td>
<td><img src="image" alt="Off-delay symbol" /> or <img src="image" alt="Off-delay symbol" /></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normally open, timed open contact (NTO). Contact is normally open when relay coil is de-energized. When relay coil is energized, contact closes instantly. When relay coil is de-energized, there is a time delay before the contact opens.</td>
<td>Normally closed, timed closed contact (NCTC). Contact is normally closed when relay coil is de-energized. When relay coil is energized, contact opens instantly. When relay coil is de-energized, there is a time delay before the contact closes.</td>
</tr>
</tbody>
</table>

**Tables**: Relay symbols used for timed contacts.
Ladder diagram **Temporized Relays or Timers**

Sequence of operation:
S1 open, TD de-energized, TD1 open, L1 off.
S1 closes, TD energizes, timing period starts, TD1 is still open, L1 is still off.
After 10 s, TD1 closes, L1 is switched on.
S1 is opened, TD de-energizes, TD1 opens instantly, L1 is switched off.

Sequence of operation:
S1 open, TD de-energized, TD1 closed, L1 on.
S1 closes, TD energizes, timing period starts, TD1 is still closed, L1 is still on.
After 10 s, TD1 opens, L1 is switched off.
S1 is opened, TD de-energizes, TD1 closes instantly, L1 is switched on.

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

**Fig. 7-4**
On-delay timer circuit (NCTO contact). (a) Operation. (b) Timing diagram.
Ladder diagram  Temporized Relays or Timers

Sequence of operation:
S1 open, TD de-energized, TD1 open, L1 off.
S1 closes, TD energizes, TD1 closes instantly, L1 is switched on.
S1 is opened, TD de-energizes, timing period starts, TD1 is still closed, L1 is still on.
After 10 s, TD1 opens, L1 is switched off.

(a)

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.

(b)

Off-delay timer circuit (NPTO contact). (a) Operation. (b) Timing diagram.

Fig. 7-5

Off-delay timer circuit (NPTC contact). (a) Operation. (b) Timing diagram.

Fig. 7-6
Ladder diagram

**Temporized Relays or Timers (PLC)**

<table>
<thead>
<tr>
<th>%TMi</th>
<th>IN</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE: <strong>TON</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TB: 1mn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM.P: 9999</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODIF: Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Characteristics:

**Identifier**: %TMi 0..63 in the TSX37

**Input**: IN to activate

**Mode**:
- **TON** On delay
- **TOF** Off delay
- **TP** Monostable

**Time basis**: TB 1mn (def.), 1s, 100ms, 10ms

**Programmed value**: %TMi.P 0...9999 (def.)

**Actual value**: %TMi.V 0...TMi.P (can be read or tested)

**Modifiable**: Y/N can be modified from the console
Ladder diagram

**Temporized Relays** or **Timers (PLC)**

**TON mode**

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

**Temporized Relays or Timers (PLC)**

**TOF mode**

App. example: turn off stairways lights after N sec the lights’ button has been released.
Temporized Relays or Timers (PLC)

**TP mode**

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

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

Timers in the Allen-Bradley PLC-5

Two alternative representations
Ladder diagram

Timers implementation in the Allen-Bradley PLC-5:

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

- 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.

Accumulated value in BCD form
Ladder diagram

Timers operation in the Allen-Bradley PLC-5

Fig. 7-9
On-delay timer sequence.

Fig. 7-10
On-delay timer instruction.

EN = Enable Bit
TT = Timer-Timing Bit
DN = Done Bit
Ladder diagram

Example of timer on-delay

Fig. 7-10

On-delay timer instruction.

<table>
<thead>
<tr>
<th>Timer element</th>
<th>EN</th>
<th>TT</th>
<th>DN</th>
<th></th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>EN TT DN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preset value</td>
<td>PRE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accumulated</td>
<td>ACC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Addressable bits
EN = Bit 15 enable
TT = Bit 14 timer timing
DN = Bit 13 done

Addressable words
PRE = Preset value
ACC = Accumulated value

(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.

(a) Ladder diagram

(b) Timing diagram
Ladder diagram

Example of a timer on-delay that sets an output after a count-down

(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 seconds later the main drive motor starts.

---

*Automatic sequential control system.*
Ladder diagram

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

![Ladder diagram of off-delay programmed timer](image)
Ladder diagram

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

Off-delay timer instructions programmed to switch motors off at 5-s intervals.
Ladder diagram

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

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

Timers

Animated demonstration:
Ladder diagram

Retentive Timers

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

Enable bit 052-17

Reset value

AC value retained when rung condition goes false

Accumulated value

Timed bit 052-15

Output lamp 011-04

Reset switch 113-07

Time in seconds

(b) Timing chart

Electromechanical retentive timer.

Switch-off region

Switch-on region

Motor-driven cam

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

Fig. 7-20

Chap. 3 - PLC Programming languages
Ladder diagram

Example of *retentive timers*

Fig. 7-22
Retentive on-delay alarm program.
Ladder diagram

Retentive Timers

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

Example:

• SW ON to start operation

• Before motor starts, lubricate 10 s with oil.

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

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

• Reset available after servicing.
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)

Annunciator flasher program.
Ladder diagram

Timers for very long time intervals

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

Example of a semaphore

Fig. 7-27
Control of traffic lights in one direction.
Example of a semaphore in both directions

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

<table>
<thead>
<tr>
<th>Red = north/south</th>
<th>Green = north/south</th>
<th>Amber = north/south</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green = east/west</td>
<td>Amber = east/west</td>
<td>Red = east/west</td>
</tr>
</tbody>
</table>

(b) Timing chart

**Fig. 7-28 (continued)**

Control of traffic lights in two directions.
Example of a semaphore in both directions.
Ladder diagram

Counters

Some applications...

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

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

Internal structure representation
Ladder diagram

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

Two alternative representations:

**Coil-formatted** counter and reset instructions

**Block-formatted** counter instruction
Ladder diagram

**Up-counters**

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

PB1 increments counting
PB2 resets the counting
Ladder diagram

Example:
Counting parts
1. Start conveyor motor
2. Passing cases increment counter
3. After 50 cases, stop motor
Chap. 3 - PLC Programming languages

Ladder diagram

**Up/down-counters**

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

PB1 increments counting
PB2 decrements the counting
PB3 resets the counter
Ladder diagram

Up/down-counters

Example:

Finite parking garage
Ladder diagram

*Cascaded Counters*

Example:

Fig. 8-21
Counting beyond the maximum count.
Ladder diagram

*Cascaded Counters*

Example:

24 hours clock

![Ladder diagram](Fig. 8-23)

A 24-h clock program.
Ladder diagram

**Cascaded Counters**

**Example:**

Memory time of event

*Internal relay OFF* stops clock
Ladder diagram

**Incremental Encoder**

counter measures **rotation angle** or **rotation speed** (if divided by time)
Ladder diagram

Incremental Encoder

Example: counter as a "length sensor"
Ladder diagram

Example with counters and timers (cont.):

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.
Ladder diagram

Example with counters and timers (cont.):

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.

Automatic stacking program.
Ladder diagram

Example with counters and timers (cont.):

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.
Ladder diagram

Example with counters and timers (cont.):

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:
### Counters in PL7

**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

Counters in Unity Pro

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

CV ≥ PV => Q:=1

R=1 => CV:=0

NOTE: counters are saturated such that no overflow occurs
Ladder diagram

Numerical Processing

Algebraic and Logic Functions
Ladder diagram

Numerical Processing

Arithmetic Functions

<table>
<thead>
<tr>
<th>+</th>
<th>addition of two operands</th>
<th>SQRT</th>
<th>square root of an operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>subtraction of two operands</td>
<td>INC</td>
<td>incrementation of an operand</td>
</tr>
<tr>
<td>*</td>
<td>multiplication of two operands</td>
<td>DEC</td>
<td>decrementation of an operand</td>
</tr>
<tr>
<td>/</td>
<td>division of two operands</td>
<td>ABS</td>
<td>absolute value of an operand</td>
</tr>
<tr>
<td>REM</td>
<td>remainder from the division of 2 operands</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operands

<table>
<thead>
<tr>
<th>Type</th>
<th>Operand 1 (Op1)</th>
<th>Operand 2 (Op2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexable words</td>
<td>%MW</td>
<td>%MW, %KW, %Xi.T</td>
</tr>
<tr>
<td>Non-indexable words</td>
<td>%QW, %SW, %NW, %BLK</td>
<td>Imm. Val., %IW, %QW, %SW, %NW, %BLK, Num. expr.</td>
</tr>
<tr>
<td>Indexable double words</td>
<td>%MD</td>
<td>%MD, %KD</td>
</tr>
<tr>
<td>Non-indexable double words</td>
<td>%QD, %SD</td>
<td>Imm. Val., %ID, %QD, %SD, Numeric expr.</td>
</tr>
</tbody>
</table>
Ladder diagram

Numerical Processing

Example:

Arithmetic functions

Use of a system variable:

%S18 – flag de overflow
Ladder diagram

Numerical Processing

<table>
<thead>
<tr>
<th>Logic Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>AND (bit by bit) between two operands</td>
</tr>
<tr>
<td>OR</td>
<td>logical OR (bit by bit) between two operands</td>
</tr>
<tr>
<td>XOR</td>
<td>exclusive OR (bit by bit) between two operands</td>
</tr>
<tr>
<td>NOT</td>
<td>logical complement (bit by bit) of an operand</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Operands 1 and 2 (Op1 and Op2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexable words</td>
<td><code>%MW, %KW, %Xi, T</code></td>
</tr>
<tr>
<td>Non-indexable words</td>
<td><code>Imm.val., %IW, %QW, %SW, %NW, %BLK, Numeric Expr.</code></td>
</tr>
<tr>
<td>Indexable double words</td>
<td><code>%MD, %KD</code></td>
</tr>
<tr>
<td>Non-indexable double words</td>
<td><code>Imm.val., %ID, %QD, %SD, Numeric expr.</code></td>
</tr>
</tbody>
</table>
Ladder diagram

Numerical Processing

Example:

Logic functions
**Ladder diagram**

**Numerical Processing**

**Priorities on the execution of the operations**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Instruction to an operand</td>
</tr>
<tr>
<td>2</td>
<td>*,,.,REM</td>
</tr>
<tr>
<td>3</td>
<td>+,-</td>
</tr>
<tr>
<td>4</td>
<td>&lt;,&gt;,&lt;=,&gt;=</td>
</tr>
<tr>
<td>5</td>
<td>=,&lt;&gt;</td>
</tr>
<tr>
<td>6</td>
<td>AND</td>
</tr>
<tr>
<td>7</td>
<td>XOR</td>
</tr>
<tr>
<td>8</td>
<td>OR</td>
</tr>
</tbody>
</table>
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 improve the legibility of the proposed solution.
Ladder diagram

Structures for Control of Flux

Halt

Stops all processes!

Events masking

MASKEVT()
Ladder diagram

There are other advanced instructions (see manual)

• Monostable

• Registers of 256 words (LIFO ou FIFO)

• DRUMs

• Comparators

• Shift-registers

... 

• Functions to manipulate floats

• Functions to convert bases and types
### Ladder diagram

#### Numerical Tables

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Maximum address</th>
<th>Size</th>
<th>Write access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal words</td>
<td>Simple length</td>
<td>%MWi:L</td>
<td>i+L&lt;=Nmax (1)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Double length</td>
<td>%MWDi:L</td>
<td>i+L&lt;=Nmax-1 (1)</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Floating point</td>
<td>%MFi:L</td>
<td>i+L&lt;=Nmax-1 (1)</td>
<td>Yes</td>
</tr>
<tr>
<td>Constant words</td>
<td>Single length</td>
<td>%KWi:L</td>
<td>i+L&lt;=Nmax (1)</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Double length</td>
<td>%KWDi:L</td>
<td>i+L&lt;=Nmax-1 (1)</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Floating point</td>
<td>%KFi:L</td>
<td>i+L&lt;=Nmax-1 (1)</td>
<td>No</td>
</tr>
<tr>
<td>System word</td>
<td>Single length</td>
<td>%SW50:4 (2)</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![Ladder diagram](image)
### Ladder diagram

#### System information: system bits

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

See manual for the remaining 100 bits generated...
## Ladder diagram

### System information: system words

<table>
<thead>
<tr>
<th>Words</th>
<th>Function</th>
<th>Description</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>%SW0</td>
<td>Master task scanning period</td>
<td>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.</td>
<td>User</td>
</tr>
<tr>
<td>%SW1</td>
<td>Fast task scanning period</td>
<td>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.</td>
<td>User</td>
</tr>
<tr>
<td>%SW8</td>
<td>Acquisition of task input monitoring</td>
<td>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.</td>
<td>User</td>
</tr>
<tr>
<td>%SW9</td>
<td>Monitoring of task output update</td>
<td>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.</td>
<td>User</td>
</tr>
<tr>
<td>%SW10</td>
<td>First cycle after cold start</td>
<td>If the bit for the current task is on 0, this indicates that the first cycle is being carried out after a cold start.</td>
<td>System</td>
</tr>
<tr>
<td>%SW11</td>
<td>Watchdog duration</td>
<td>Reads the duration of the watchdog as set in configuration. It is expressed in ms (10...500 ms).</td>
<td>System</td>
</tr>
</tbody>
</table>

See manual for the remaining 140 words generated...
A program can be built from:
Tasks, that are executed cyclically or periodically.

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

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

*Unity - Project Browser*
### Ladder diagram

#### Software Organization

**MAST – Master Task Program**
- Composed by sections
- Execution **Cyclic or Periodic**

<table>
<thead>
<tr>
<th>SAS (LD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oven1 (GRAFCET)</td>
</tr>
<tr>
<td>PRL (LD)</td>
</tr>
<tr>
<td>Chart</td>
</tr>
<tr>
<td>POST (IL)</td>
</tr>
<tr>
<td>Drying (LD)</td>
</tr>
<tr>
<td>Cleaning (IL)</td>
</tr>
</tbody>
</table>

---

**Properties of MAST**

![Properties of MAST](image)
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

**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 transitions
- Counters
- Upon telegrams reception
- %S38 Enables or disables event processes

(also with MASKEVT() or UNMASKEVT())
Ladder diagram Development tools

Each PLC has limitations in terms of connections

Example:

![Ladder diagram image]

Fig. 5-27

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

It is important to learn the potentialities and ... the limitations of the developing tools, i.e. STUDYING the manuals is a MUST.
Last but not least, *learn how to develop and debug programs* (and how to do some fine tuning).

![Ladder diagram](image)

### Development tools

<table>
<thead>
<tr>
<th>Channel</th>
<th>Symbol</th>
<th>State</th>
<th>Error</th>
<th>Fallback</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td>STOP</td>
<td>ALARM</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>F1</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>F0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>F0</td>
<td>ERR</td>
<td>STOP</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td>STOP</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>0</td>
<td>ERR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Forcing**
- F4: Force to 0
- F5: Force to 1
- F6: Unforce

**Write**
- F7: Set
- F8: Reset
Last but not least, **learn how to develop and debug programs** (and how to do some fine tuning).