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]
Standard languages (IEC-61131-3):
Ladder Diagram; Instruction List, and Structured Text.
Software development resources.

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

**Ladder Diagram**

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

**Structured Text**

```plaintext
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)**

- 1
- 2
- 3
- 4
- Direita
- Carrega
- Esquerda

---

Page 3
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:

```
<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>%I0.3.0</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.1</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.2</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.3</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.4</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.5</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.6</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.7</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.8</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.9</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.10</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.11</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.12</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.13</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.14</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%I0.3.15</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.16</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.17</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.18</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.19</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.20</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.21</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.22</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.23</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.24</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
<tr>
<td>%Q0.3.25</td>
<td></td>
<td>EBOOL</td>
<td></td>
</tr>
</tbody>
</table>
```

### Ladder diagram

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

Basic Instructions

Load

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

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

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

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

### Basic Instructions

**Load operands**

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></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></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></td>
<td>LDR</td>
<td>:= RE</td>
<td>%I,%Q,%M</td>
</tr>
<tr>
<td></td>
<td>LDF</td>
<td>:= FE</td>
<td>%I,%Q,%M</td>
</tr>
</tbody>
</table>
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).

- The result of the logic function de-energizes the relay (resets the latch).
Ladder diagram

Basic Instructions

*Store* operands

<table>
<thead>
<tr>
<th>Permitted operands</th>
<th>The following table gives a list of the operands used for these instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language data</td>
<td>Instruction list</td>
</tr>
<tr>
<td>( )</td>
<td>ST</td>
</tr>
<tr>
<td>( )</td>
<td>STN</td>
</tr>
<tr>
<td>( )</td>
<td>S</td>
</tr>
<tr>
<td>( s )</td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>R</td>
</tr>
<tr>
<td>( )</td>
<td></td>
</tr>
</tbody>
</table>
Ladder diagram

*Allen Bradley notation*

Relays with *latch* and *unlatch*

Fig. 6-50

Schematic of electromagnetic latching relay.
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

**XOR**

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

<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>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>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>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>
</tbody>
</table>
Ladder diagram

Ladder assembling

![Ladder Diagram]

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:

![Ladder logic program diagram](image)
Ladder diagram

Example:

(a) Hard-wired circuit

(b) Programmed circuit

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

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

**Imbricated** (nested) contacts and **alternative** solution
Contacts in the **vertical** and **alternative** solution

![Ladder diagram image]

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:

![Fig. 5-30](Original circuit.)

![Fig. 5-31](Reprogrammed circuit.)

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

Solid-state timing relay  Pneumatic timing relay  Plug-in timing relay
**Ladder diagram**  
*Temporized Relays or Timers (pneumatic)*

The **instantaneous** contacts change state as soon as the timer coil is powered.  
The **delayed** contacts change state at the end of the time delay.
Ladder diagram  Temporized Relays or Timers

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

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

<table>
<thead>
<tr>
<th>On-delay symbols</th>
<th>Off-delay symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="On-delay symbol" /> or <img src="image2" alt="On-delay symbol" /></td>
<td><img src="image3" alt="Off-delay symbol" /> or <img src="image4" alt="Off-delay symbol" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On-delay symbols</th>
<th>Off-delay symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normally open, timed closed contact (NOTC).</td>
<td>Normally open, timed closed contact (NCTC).</td>
</tr>
<tr>
<td>Contact is open when relay coil is de-energized.</td>
<td>Contact is normally open when relay coil is de-energized.</td>
</tr>
<tr>
<td>When relay is energized, there is a time delay in closing.</td>
<td>When relay is energized, there is a time delay before the contact opens.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On-delay symbols</th>
<th>Off-delay symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="On-delay symbol" /> or <img src="image6" alt="On-delay symbol" /></td>
<td><img src="image7" alt="Off-delay symbol" /> or <img src="image8" alt="Off-delay symbol" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>On-delay symbols</th>
<th>Off-delay symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normally closed, timed open contact (NCTO).</td>
<td>Normally closed, timed open contact (NOTO).</td>
</tr>
<tr>
<td>Contact is closed when relay coil is de-energized.</td>
<td>Contact is normally closed when relay coil is de-energized.</td>
</tr>
<tr>
<td>When relay is energized, there is a time delay in opening.</td>
<td>When relay is energized, contact closes instantly.</td>
</tr>
</tbody>
</table>

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

![Diagram of temporized relays or timers](image)

**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.

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

\textbf{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.

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.

**Fig. 7-5**
Off-delay timer circuit (NOTO contact). (a) Operation. (b) Timing diagram.

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

### 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>TB: 1mn</td>
<td>TM.P: 9999</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: Timer On delay
  - TOF: Timer Off delay
  - TP: Monostable
- **Time basis:** TB 1mn (def.), 1s, 100ms, 10ms
- **Programmed value:** %TMi.P 0...9999 (def.)
  - period=TB*TMi.P
- **Actual value:** %TMi.V 0...TMi.P (can be read or tested)
- **Modifiable:** Y/N can be modified from the console

**Ladder diagram**

![Ladder diagram](image-url)
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 gives a controlled (fixed) duration pulse to start a motor.
Ladder diagram

*Timers in the Allen-Bradley PLC-5*

Two alternative representations

![Diagram of a ladder logic circuit with labels for timer address, type of timer, timer preset value, time accumulated or current value, time base of timer, contact for rung continuity, output line, control line, preset time, time base, accumulated time, reset line, and bus for formatted timer instruction.](image)
Ladder diagram

Timers implementation in the Allen-Bradley PLC-5:

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

Accumulated value in BCD form

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

- **Enabled bit.** This bit is set to 1 when timer rung conditions are true.
- **Timed Bit.** This bit is set to 1 or 0 when the timer has timed out; that is, $AC = PR$. 


**Ladder diagram**

*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>Internal use</th>
<th>Preset value</th>
<th>Accumulated value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>Word</td>
<td>0</td>
<td>1</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.

Fig. 7-11

On-delay timer.
Ladder diagram

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

Fig. 7-12
On-delay timer with instantaneous output programming.
Ladder diagram

Example of *timer on-delay*

![Ladder diagram](image)

(a) Relay ladder schematic diagram

**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.
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*

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 is reset when input switch is opened.

Input switch 113:06
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.

Cam-operated contact
Switch-off region
Switch-on region
Motor-driven cam
Motor accumulated motion (rotation) defines the on/off timing.
Ladder diagram

Example of *retentive timers*

![Ladder diagram image](image)

**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

Fig. 7-24
Sequential time-delayed motor-starting circuit.
Ladder diagram

Cascaded Timers (bistable system)
Ladder diagram

Timers for very long time intervals

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

Example of a semaphore

Control of traffic lights in one direction.
Example of a semaphore in both directions

<table>
<thead>
<tr>
<th>Light</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>30 s on</td>
</tr>
<tr>
<td>Green</td>
<td>25 s on</td>
</tr>
<tr>
<td>Amber</td>
<td>5 s on</td>
</tr>
</tbody>
</table>

(b) Timing chart

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*
Ladder diagram

Example

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

**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.
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.
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:
Ladder diagram

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 image]
Ladder diagram

Numerical Processing

Arithmetic Functions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Symbol(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition of two operands</td>
<td>%MW</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction of two operands</td>
<td>%MW,%KW,%Xi.T</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication of two operands</td>
<td>%QW,%SW,%NW,%BLK</td>
</tr>
<tr>
<td>/</td>
<td>Division of two operands</td>
<td>Imm.Val.,%IW,%QW,%SW,%NW,%BLK, Num.expr.</td>
</tr>
<tr>
<td>REM</td>
<td>Remainder from the division of 2 operands</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

Logic Functions

<table>
<thead>
<tr>
<th>Operator</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>%MW,%KW,%Xi,T</td>
</tr>
<tr>
<td>Non-indexable words</td>
<td>Imm.val.,%IW,%QW,%SW,%NW,%BLK, Numeric Expr.</td>
</tr>
<tr>
<td>Indexable double words</td>
<td>%MD,%KD</td>
</tr>
<tr>
<td>Non-indexable double words</td>
<td>Imm.val.,%ID,%QD,%SD,Numeric expr.</td>
</tr>
</tbody>
</table>
Ladder diagram

Numerical Processing

Example:

Logic functions

```
%MW10>100

%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
%M0
&M
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

\[ \%M10 \]

Stops all processes!

Events masking

\[ \%M0 \]

\[ \%M8 \]

MASKEVT() UNMASKEVT()
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>

```
%M0

%MWO:10:=%MW0:10+100

%MWO:10:=%MW0:10+100

%I3.2

%MWO:10:=%MW0:10+100

%I3.3

%MWO:10:=%MW0:10+100
```

Page 88
## 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>
</table>
| %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: ![Graph of %S4](image) | -             | YES   | YES   |
| %S5   | Time base 100ms| Idem %S4                                                                                                                                                                                                   | -             | YES   | YES   |
| %S6   | Time base 1s  | Idem %S4                                                                                                                                                                                                   | -             | YES   | YES   |
| %S7   | Time base 1mn | Idem %S4                                                                                                                                                                                                   | -             | YES   | YES   |

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>
</tbody>
</table>
| %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. It 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     |
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

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

SR0
Ladder diagram  Software Organization

**FAST – Fast Task Program**

Priority greater than MAST

- Executed Periodically (1-255ms)
- Verified by a *Watchdog*, impacts on %S11
- %S31 *Enables* or *disables* a FAST
- %S33 gives the execution time for FAST
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  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).
Last but not least, learn how to develop and debug programs (and how to do some fine tuning).