

Lab 2: Programming a GRAFCET using Ladder Logic

Lab Objective

The objective of this lab is to learn how to translate a GRAFCET into a Ladder Logic program on a Programmable Logic Controller (PLC). Through a practical example, you will learn to:

- Configure digital (discrete) inputs and outputs on the SIMATIC S7-1200 PLC.
- Properly declare the variables used in the program.
- Develop and implement a control program using Ladder Logic.

Some programmable logic controllers (PLCs) offer native support for programming using the GRAFCET language. However, in many practical situations, especially when the PLC does not support GRAFCET directly, it becomes necessary to translate the GRAFCET into another language.

In such cases, it is often useful to convert a GRAFCET into a Ladder Logic (LAD) program to ensure compatibility with the PLC being used.

Step Activation and Deactivation Rules in GRAFCET

GRAFCET steps can be considered as memory functions. Each step has an activation condition (CA_i) and a deactivation condition (CD_i).

- **Activation of a step:** A step is activated if the immediately preceding step is active and the associated transition condition is true.
- **Deactivation of a step:** A step is deactivated if the activation condition of the following step is fulfilled (i.e., becomes true).

Evolution rules for steps based on activation and deactivation conditions:

- If $CA_i = 1$ (true) and $CD_i = 0$ (false), the step is activated (**state = 1**).
- If $CA_i = 0$ and $CD_i = 1$, the step is deactivated (**state = 0**).
- If $CA_i = 0$ and $CD_i = 0$, the step retains its previous state.
- If both $CA_i = 1$ and $CD_i = 1$, the step remains active (**state = 1**), in accordance with **Rule 3** of GRAFCET.

Rule 3: When a step is simultaneously activated and deactivated, it remains active.

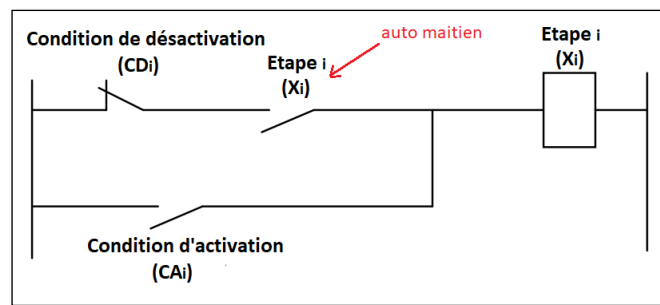
The following truth table summarizes all possible combinations of the activation condition (CA_i) and the deactivation condition (CD_i), along with their impact on the future state of a step x_i .

Truth Table for Step Activation/Deactivation Logic

N°	x_i (current)	CD_i	CA_i	x_i (next)	Observation
0	Inactive	0	0	0	Step state maintained
1		0	1	1	Activation of $x_i : 0 \rightarrow 1$
2		1	0	0	Step state maintained
3		1	1	1	Activation of $x_i : 0 \rightarrow 1$ (Activation dominates)
4	Active	1	0	1	Step state maintained
5		1	1	1	Step state maintained
6		0	1	0	Deactivation of $x_i : 1 \rightarrow 0$
7		0	0	0	Step state maintained

Based on the analysis of the truth table, we derived the following equation:

$$X_i = \overline{CD_i} \cdot X_i + CA_i \quad (1)$$



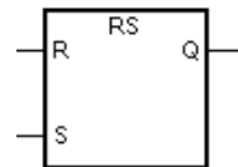
We observe that the previously presented truth table, which describes the behavior of each GRAFCET step, closely resembles that of an SR flip-flop. The only difference lies in the case where both inputs are simultaneously active ($R = S = 1$), which may lead to an undefined or unstable output state.

However, to make the two equations identical, the SR flip-flop can be replaced with an RS1 flip-flop, which forces the output to 1 when both inputs are active at the same time. This analogy enables a simple and efficient implementation of GRAFCET steps using RS1 flip-flops in PLC programming.

Reminder: Truth Table of the SR Flip-Flop

Q_n (Actuel)	R	S	Q_{n+1} (Future)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	-
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	-

Indeterminate state



Symbol of the RS Flip-Flop

Output equation of the SR flip-flop

$$Q_{i+1} = \overline{R} \cdot Q_i + S \quad (2)$$

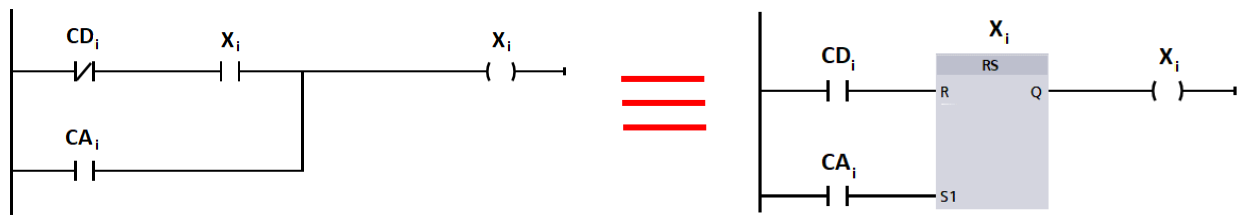
Truth Table of the RS1 Flip-Flop

Q_n (Actuel)	R	S	Q_{n+1} (Future)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



Symbol of the RS1 Flip-Flop

Representation of a GRAFCET step using an RS1 flip-flop

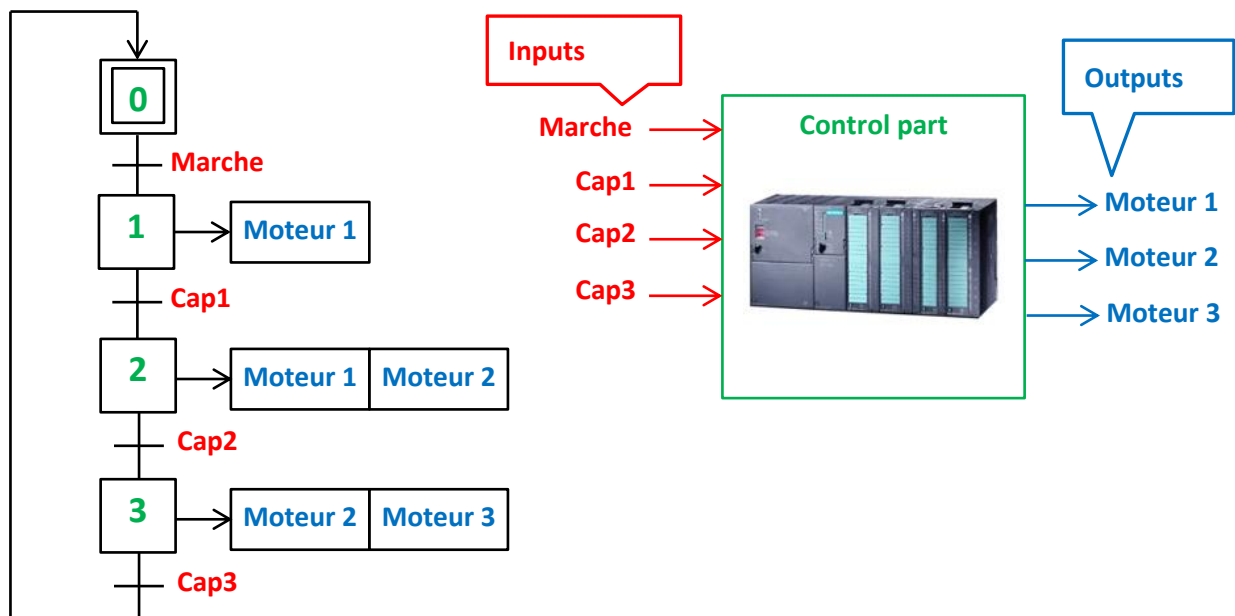


Example 1:

An automated system is represented by the GRAFCET diagram below, consisting of 4 steps (X_0 , X_1 , X_2 , and X_3).

The system has 4 digital inputs (discrete inputs): (Marche, Cap1, Cap2 et Cap3).

It also has 3 outputs: Motor 1, Motor 2, and Motor 3.



To implement the GRAFCET shown above, it is necessary to associate each input and output with a physical address on the corresponding input and output modules. In addition, each GRAFCET step will be clearly represented using a memory marker.

Note: A memory marker refers to an internal memory bit.

Input Assignment Table

Receptivity	Input on the module
Marche	I0.0
Cap1	I0.1
Cap2	I0.2
Cap3	I0.3

Output Assignment Table

Actuator	Sortie sur le module
Moteur 1	Q0.0
Moteur 2	Q0.1
Moteur 3	Q0.2

Step Assignment Table

Step	Memento
X_0	M0.0
X_1	M0.1
X_2	M0.2
X_3	M0.3

Notation:

I: is used to designate an input, Q: is used to designate an output, M: is used to designate a memory bit (marker)

Actuator Activation Table

Actuator	Activated in Step (s)
Moteur 1	$X_1 + X_2$
Moteur 2	$X_2 + X_3$
Moteur 3	X_3

Step Activation and Deactivation Table

Step (X_i)	Activation condition: (CA_i)	Deactivation condition: (CD_i)
X_0	$X_3 \cdot \text{Cap3}$	X_1
X_1	$X_0 \cdot \text{Marche}$	X_2
X_2	$X_1 \cdot \text{Cap1}$	X_3
X_3	$X_2 \cdot \text{Cap2}$	X_0

Standard Variables Table (Image of the table in the TIA (Totally Integrated Automation) Portal V16 software)

Before anything else, you must start by declaring the variables (see figure below).


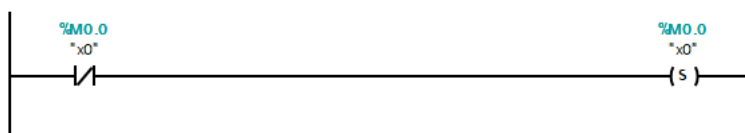


Table de variables standard			
	Nom	Type de données	Adresse
1	x0	Bool	%M0.0
2	x1	Bool	%M0.1
3	x2	Bool	%M0.2
4	ca0	Bool	%M0.3
5	ca1	Bool	%M0.4
6	ca2	Bool	%M0.5
7	marche	Bool	%I0.0
8	cap 1	Bool	%I0.1
9	cap 2	Bool	%I0.2
10	moteur 1	Bool	%Q0.0
11	moteur 2	Bool	%Q0.1
12	moteur 3	Bool	%Q0.2
13	x3	Bool	%M0.6
14	ca3	Bool	%M0.7
15	cap 3	Bool	%I0.3

The GRAFCET must be initialized, meaning that its initial step must be activated (in our case, setting X_0 to 1). This operation must be carried out at startup, that is, executed once when the CPU transitions from **STOP** to **RUN** mode.

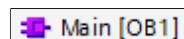
Siemens PLCs offer a startup organization block called **OB100**, which is executed only once during the transition from STOP to RUN. (See the network below)

Startup Program: **Startup Block (OB100)**

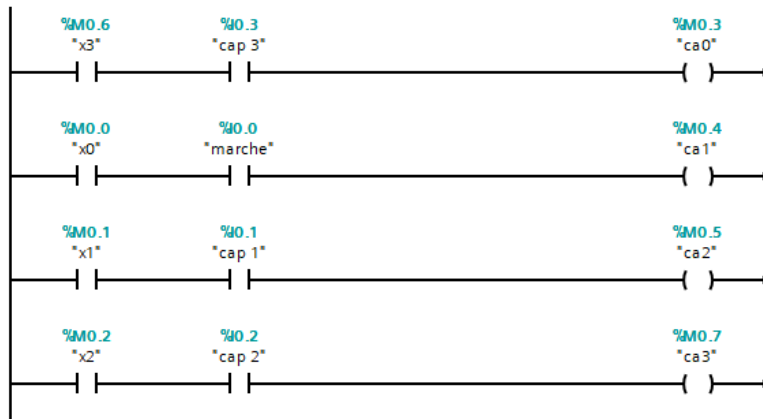


In a PLC, the main program is executed cyclically. Siemens PLCs use the **Organization Block OB1** to represent the main program.

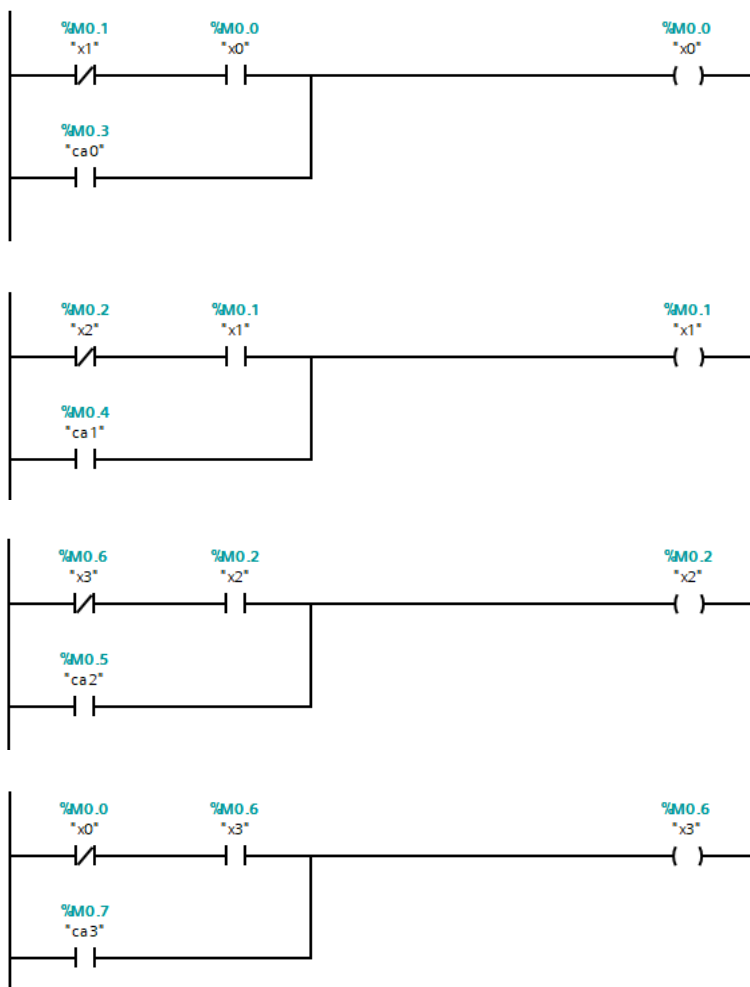
Main Program: **Main Organization Block (OB1)**



Les conditions d'activation des étapes



After defining the activation conditions for each step.



We use the networks below to activate the system outputs (moteur 1, moteur 2 et moteur 3).

