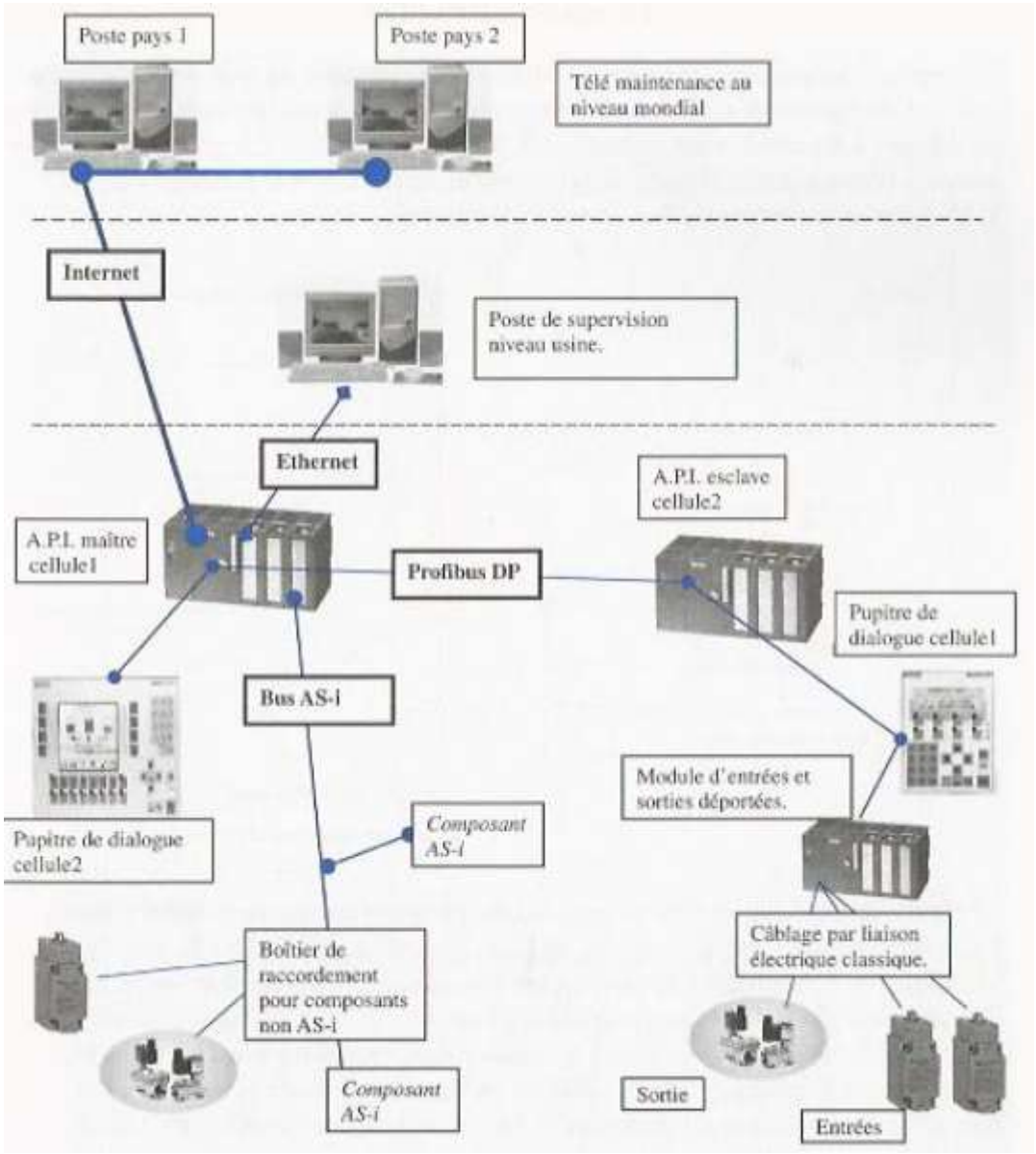
Asean-factori

2. PLC : Programmable Logic Controller

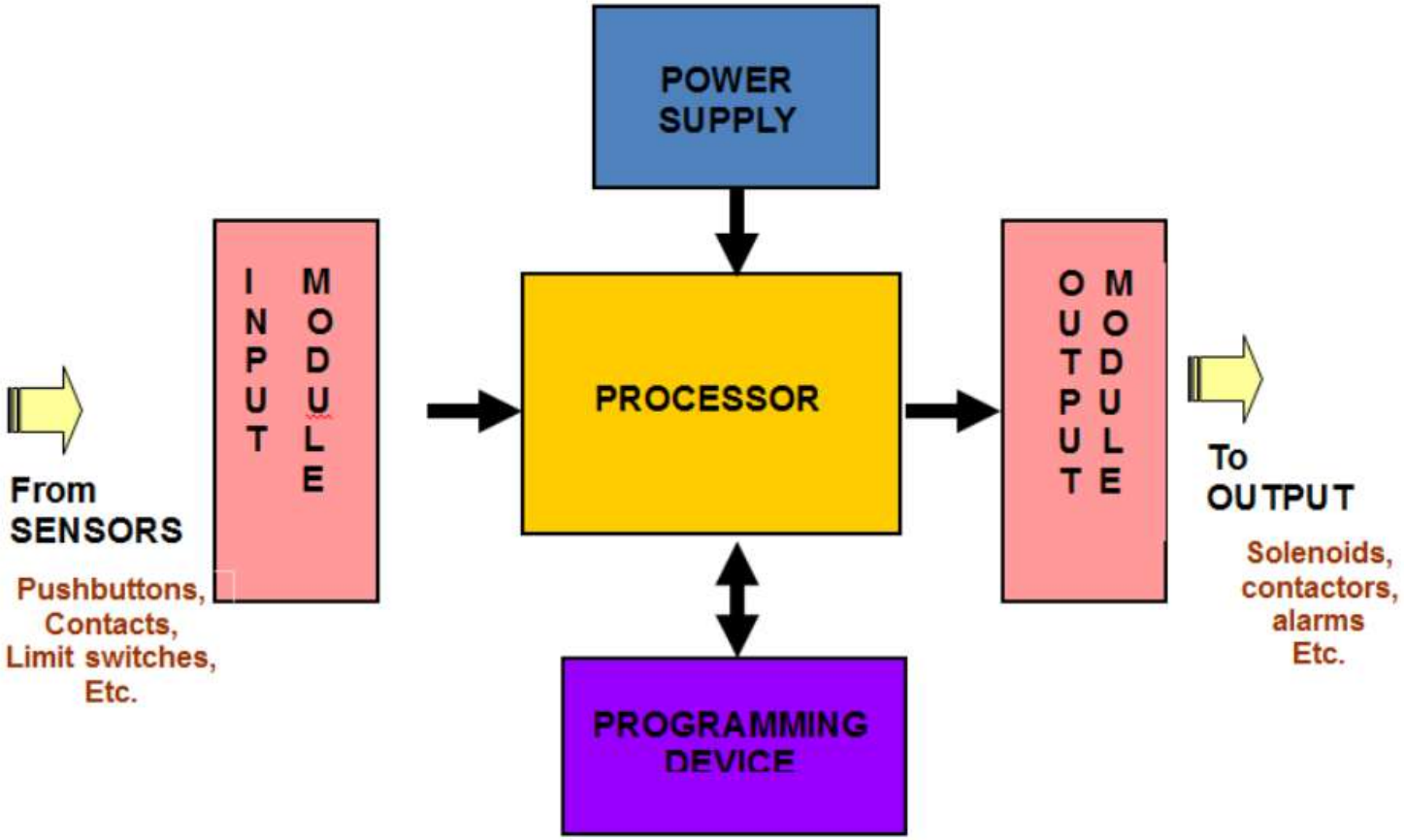
Jean-Marc THIRIET

jean-marc.thiriet@univ-grenoble-alpes.fr

http://www.gipsa-lab.grenoble-inp.fr/~jean-marc.thiriet/asean/asean.html


Poste pays 1
Poste pays 2
Télé maintenance au niveau mondial
Internet
Poste de supervision niveau usine.
Ethernet
A.P.I. esclave cellule2
A.P.I. maître cellule1
Profibus DP
Pupitre de dialogue cellule1
Bus AS-i
Module d'entrées et sorties déportées.
Pupitre de dialogue cellule2
Composant AS-i
Câblage par liaison électrique classique.
Boîtier de raccordement pour composants non AS-i
Composant AS-i
Sortie
Entrées

Structure of a PLC (1/3)
Functional View



Structure of a PLC (2/3)

- Processor (CPU, Central Processing Unit): the microprocessor interprets the input signals and performs the control actions according to the program stored in memory, communicating the decisions to the outputs in the form of action signals
- Power supply unit: indispensable, converts an AC voltage into the low DC voltage necessary for the processor and the I/O modules
- Programming device: used to enter the program in the processor memory. The pgm is developed on the device, then transferred to the PLC memory
- The memory contains the program that defines the control actions performed by the microprocessor. It also contains the data from the inputs for processing, as well as the data from the outputs.

Structure of a PLC (3/3)

- Input/Output (I/O) interfaces
  - Receive and send information to external devices
  - Inputs: switches, sensors...
  - Outputs: coils, valves...
  - I/O signals: discrete (binary), digital, analog
- Communication interfaces
  - Receive and transmit data over the communication networks that link the PLC to other PLCs or remote devices
  - Synchronization

The first PLC, model 084, was invented by Dick Morley in 1969

The "084" - Details

The **"084"** consisted of three major components mounted on two vertical rails, one of which was hinged to allow for service access to the front and back.

Ladder Logic:

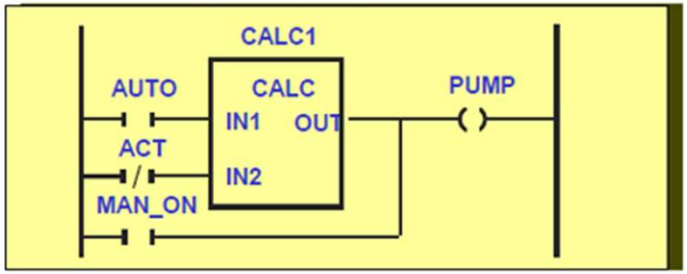
The use of **Ladder Logic** was significant in the rapid acceptance of the **"084"** because the very same engineers and electricians who designed and maintained Factory Automation Systems could also program an **"084"**. Ladder Logic was simply an electronic version of the elementary electrical diagram that they already used -- not the case for other types of control systems being designed at the time.

PLC Languages: IEC 61131

Instruction List (IL)

```
A: LD    %IX1  (* PUSH BUTTON *)
   ANDN  %MX5  (* NOT INHIBITED *)
   ST    %QX2  (* FAN ON *)
```

Ladder Diagram (LD)

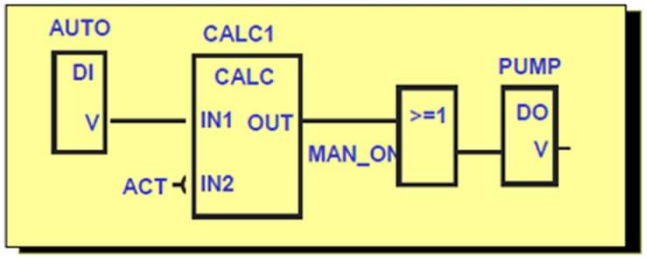


Structured Text (ST)

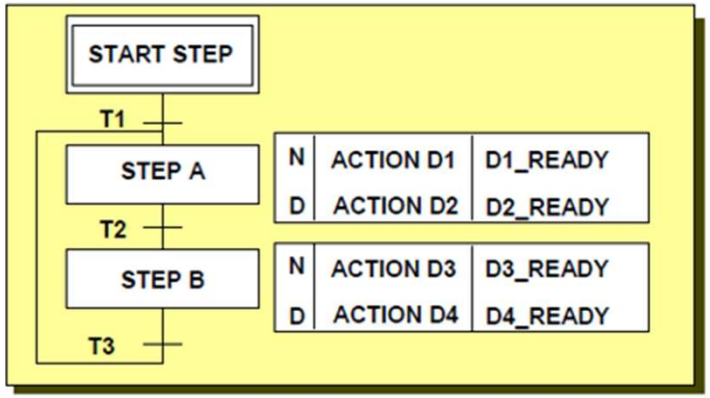
```
VAR CONSTANT X : REAL := 53.8 ;
Z : REAL; END_VAR
VAR aFB, bFB :  FB_type; END_VAR

bFB(A:=1, B:='OK');
Z := X - INT_TO_REAL (bFB.OUT1);
IF Z>57.0 THEN aFB(A:=0, B:="ERR");
ELSE  aFB(A:=1, B:="Z is OK");
END_IF
```

Function Block Diagram (FBD)



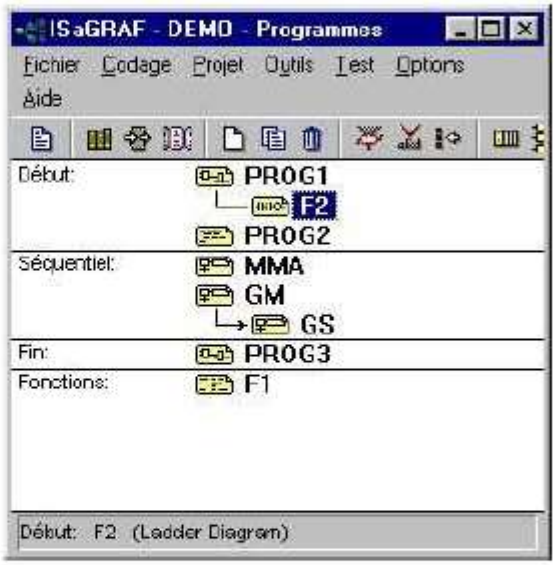
Sequential Flow Chart (SFC)



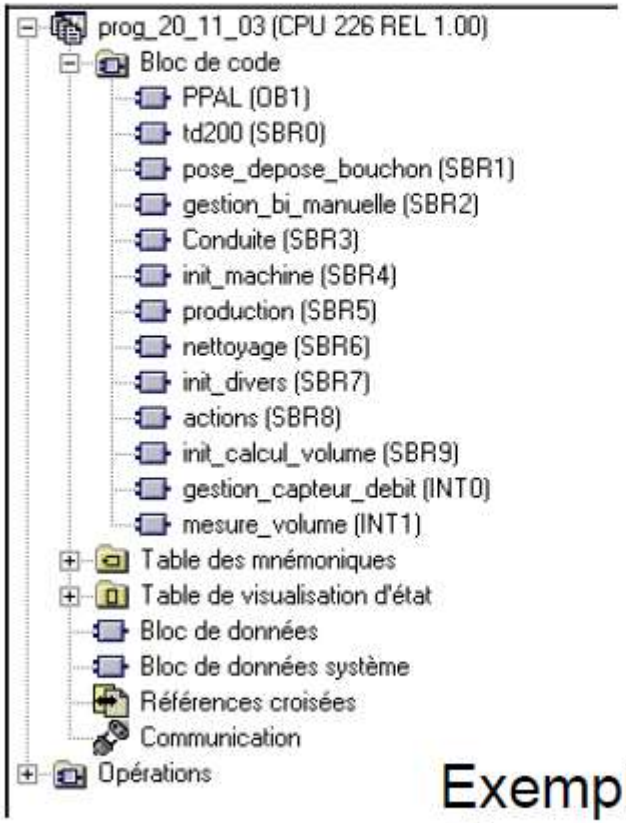
Comparaison des langages

LANGAGE	AVANTAGES	INCONVENIENTS
LD	facile à lire et à comprendre par la majorité des électriciens langage de base de tout PLC	suppose une programmation bien structurée
FBD	Très visuel et facile à lire	Peut devenir très lourd lorsque les équations se compliquent
ST	Langage de haut niveau (langage pascal) Pour faire de l'algorithmique	Pas toujours disponible dans les ateliers logiciels
IL	langage de base de tout PLC type assembleur	très lourd et difficile à suivre si le programme est complexe Pas visuel.
SFC	Description du fonctionnement (séquentiel) de l 'automatisme. Gestion des modes de marches Pas toujours accepté dans l'industrie…	Peu flexible

Multi-langages, multi-programmes !



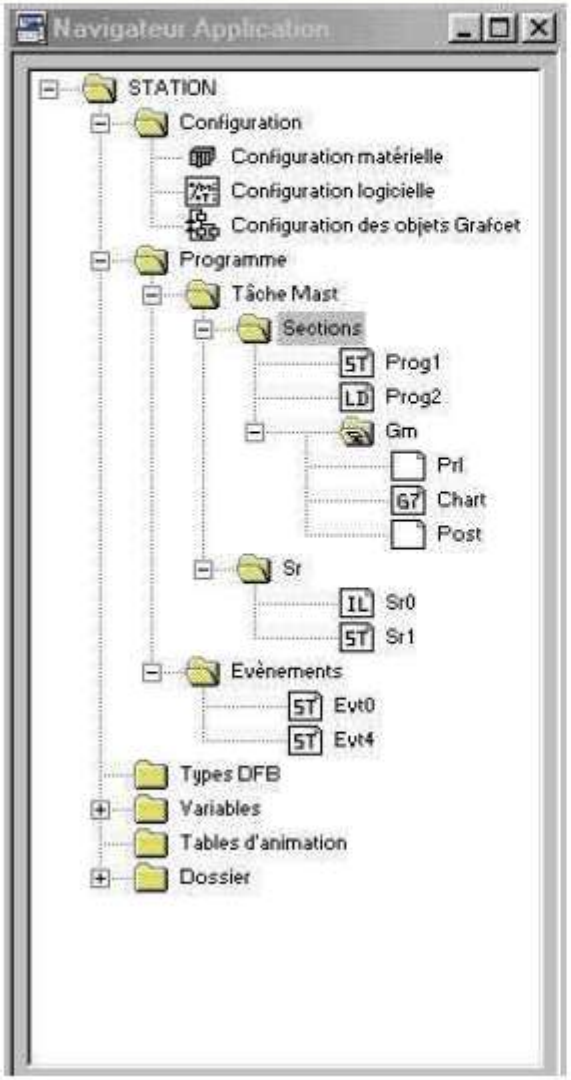
Exemple
Isagraf



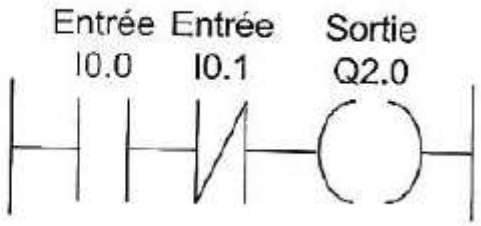
Exemple
Siemens

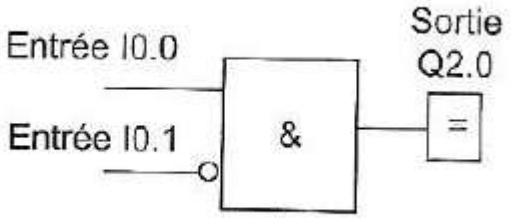
Multi-langages, multi-programmes !

Exemple
Schneider



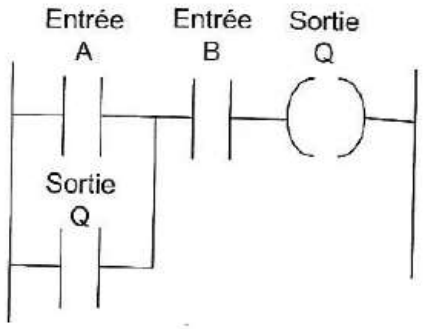
Equivalence between Ladder diagrams (LD) and Function Block diagrams (FBD)

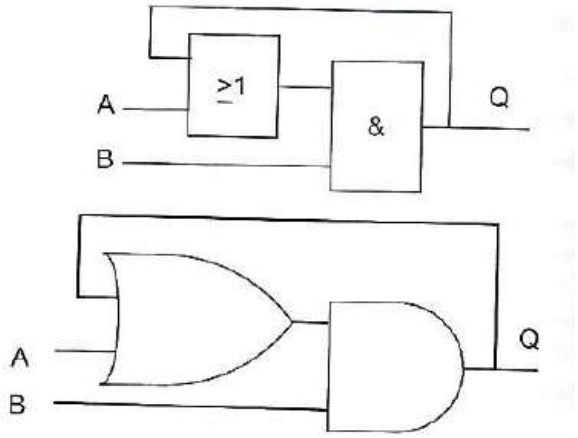




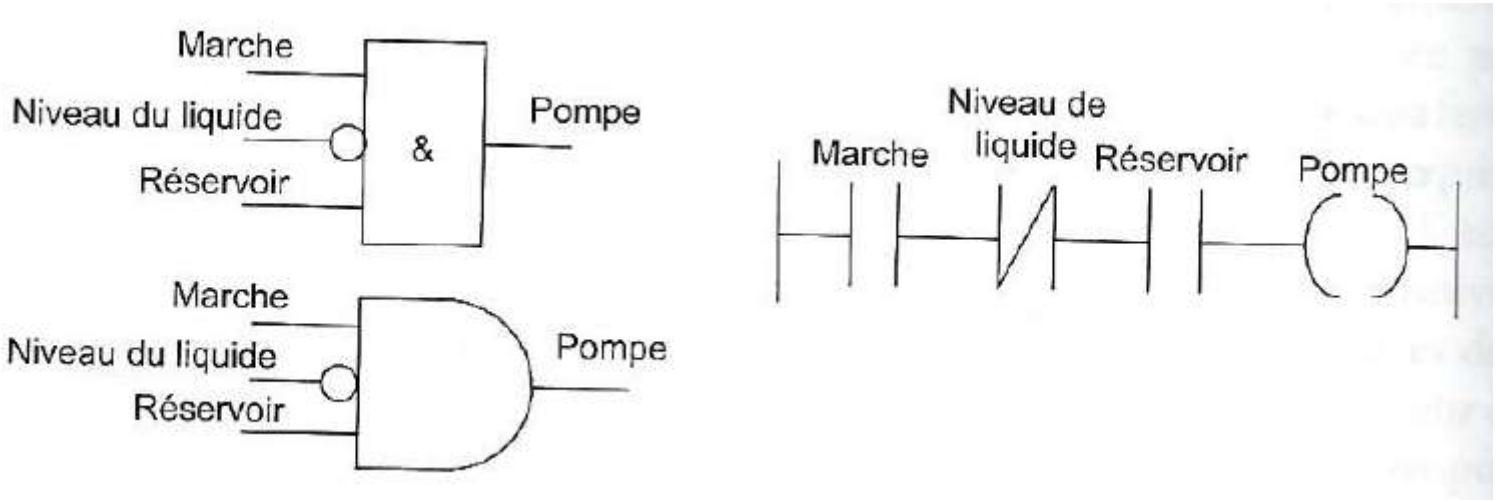
Latching circuit:
The output Q is fed back as an input (this is a "relay" that stores the state of Q)

Circuit à verrouillage

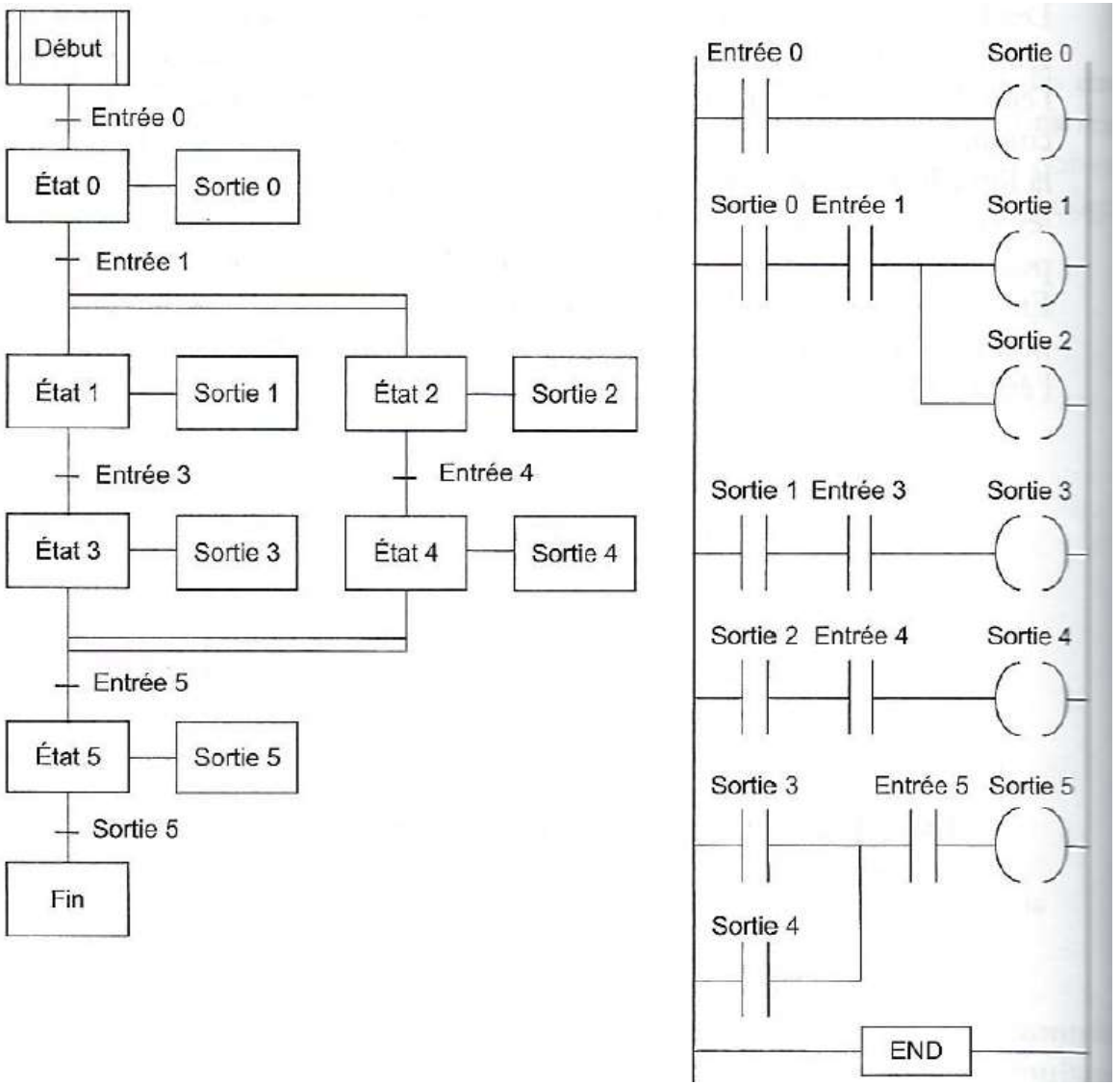




Equivalence between Ladder diagrams (LD) and Function Block diagrams (FBD)

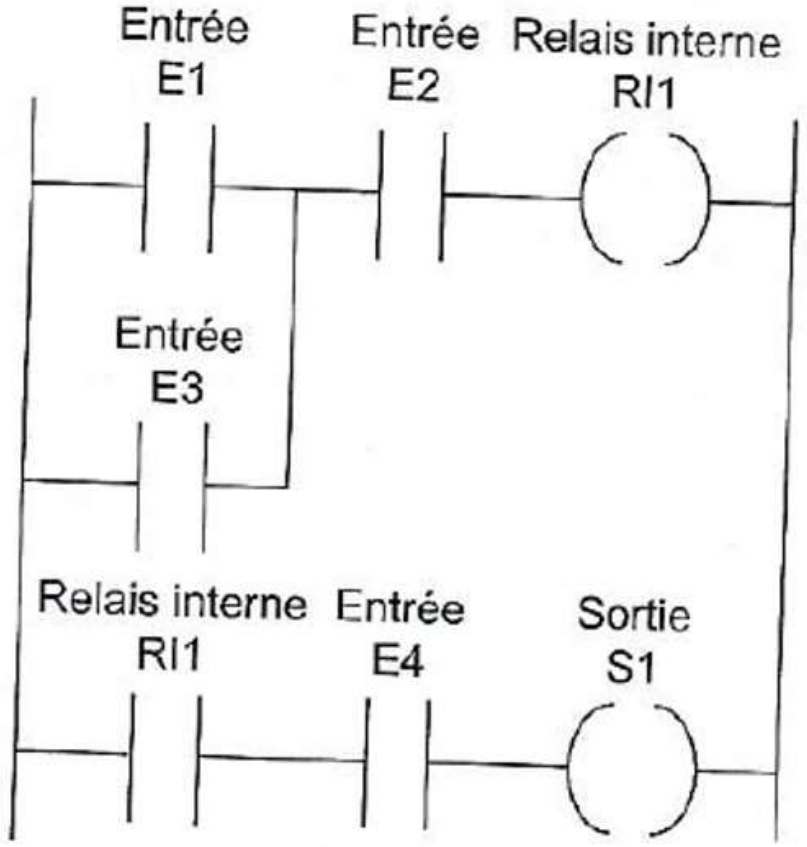


Equivalence between Ladder diagrams (LD) and Sequential Flow Chart (SFC)

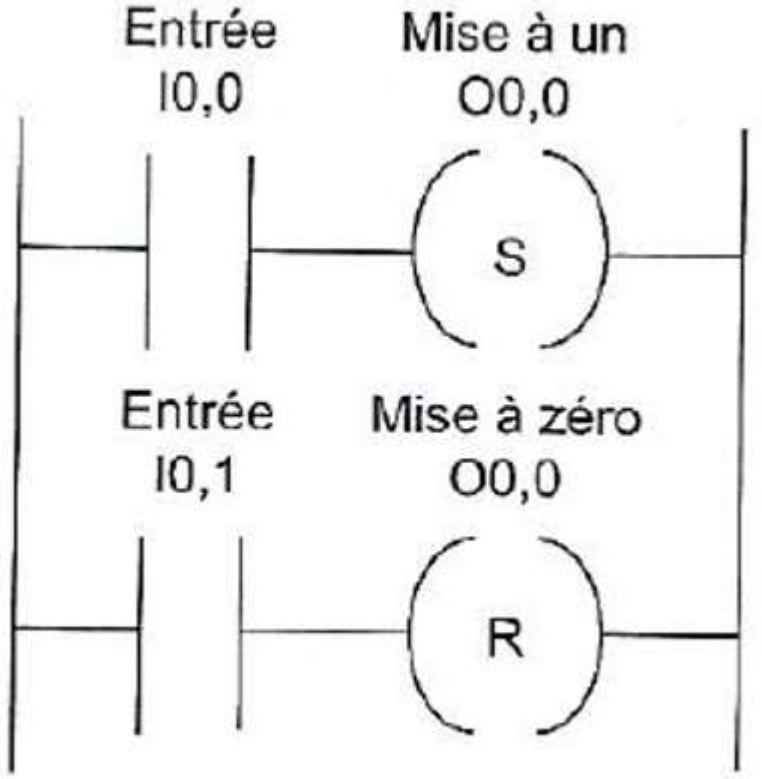


Internal Relay

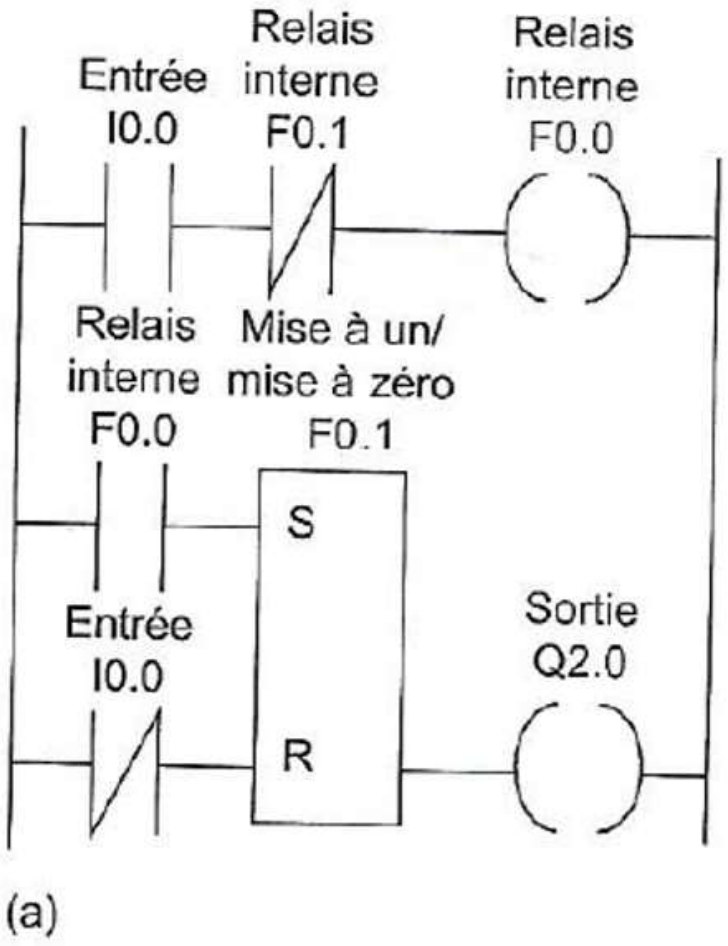
- An internal relay allows to keep (memorize) a value (bit)
- It is equivalent to a memory
- It is <> from an I/O
- In Siemens, it can be managed as a Flag (F)



Setting to 0 and setting to 1 of an internal relay



Flip-flop solution (FR : bascule)



(a)

Les constructeurs

Schneider Electric

WAGO

ABB

Honeywell

PHOENIX CONTACT

When good enough just isn't good enough
Weidmüller

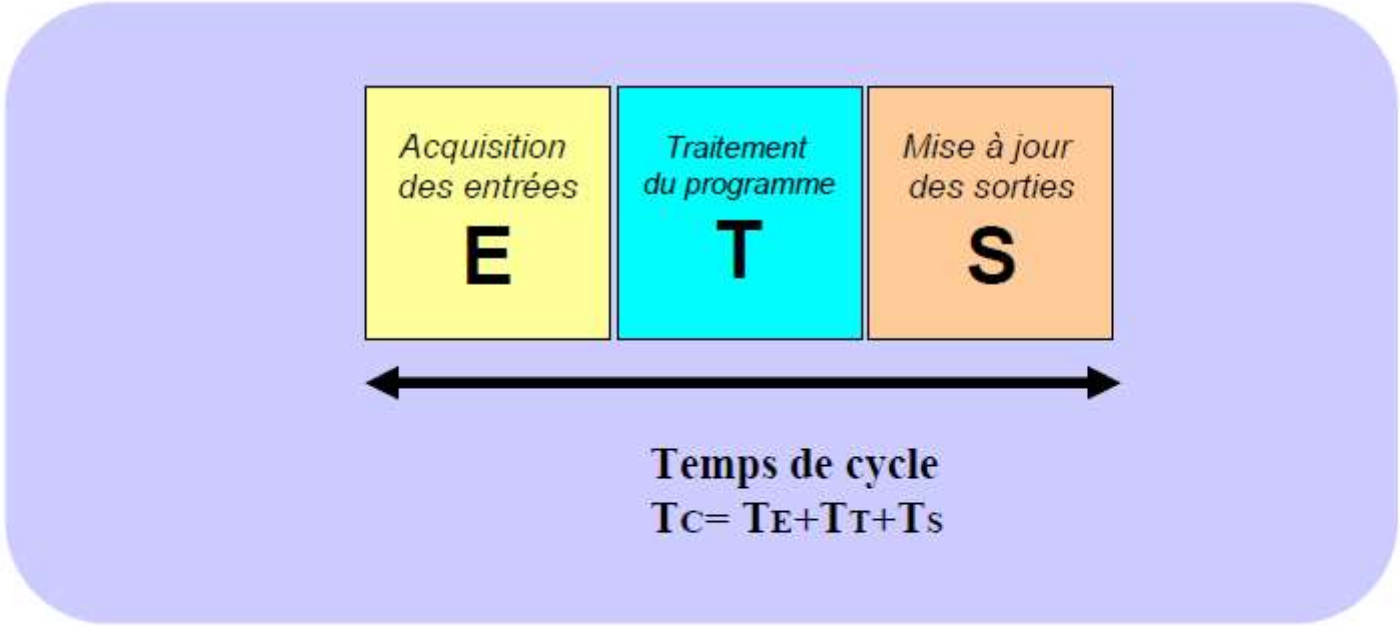
Allen-Bradley

Rockwell Automation

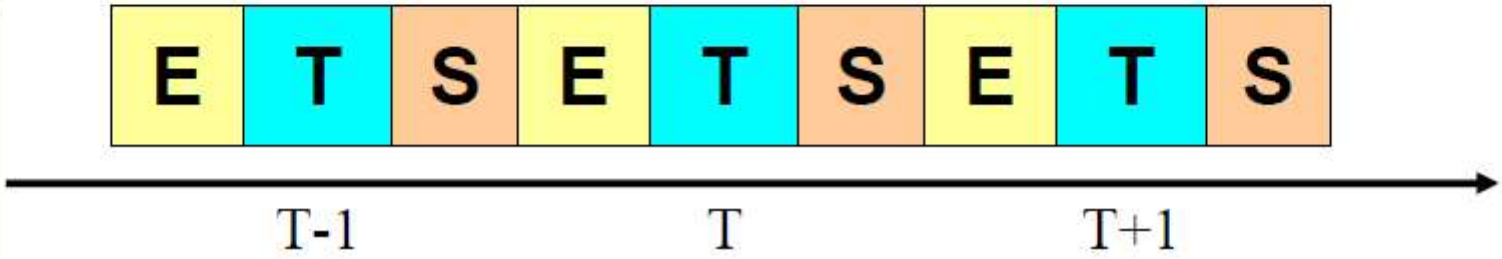
HITACHI
Inspire the Next

NAiS Matsushita Electric Works-
Automation Controls Company

Tâche Automate

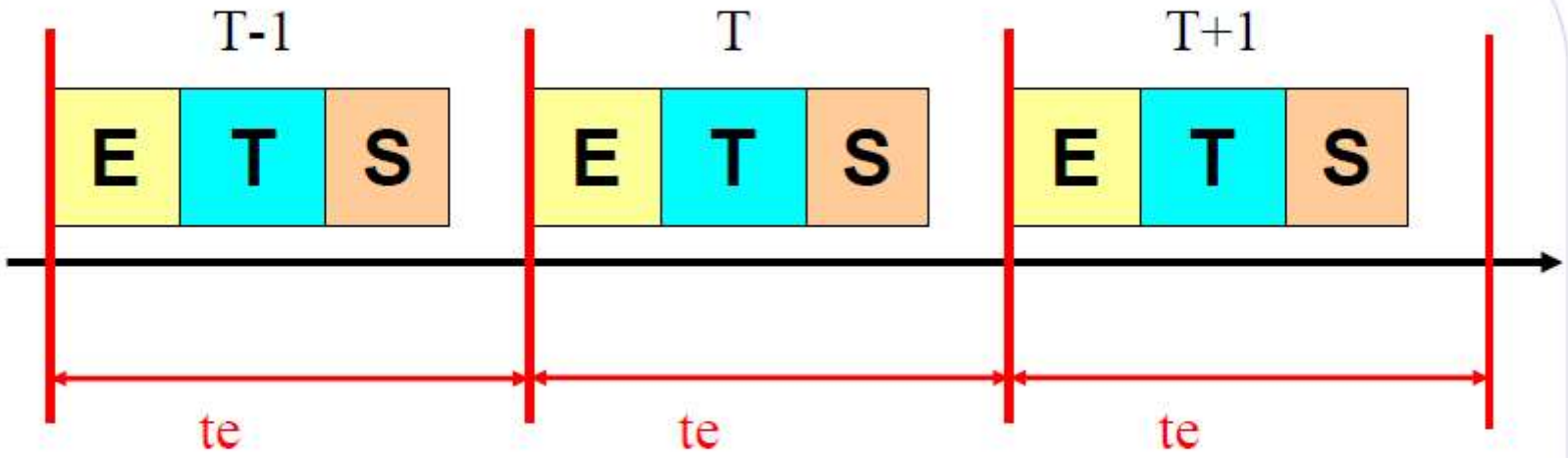


fonctionnement mono-tâche cyclique (asynchrone)



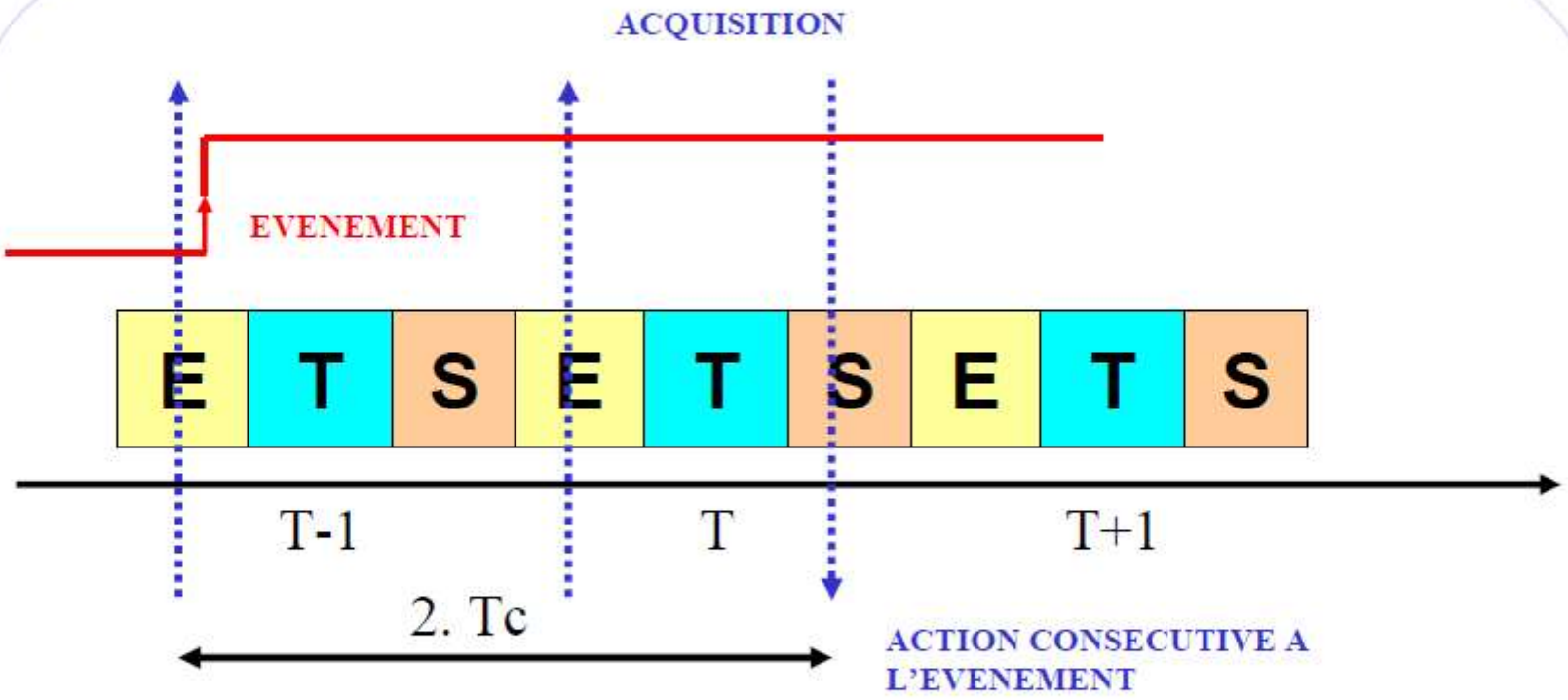
Ce type de fonctionnement consiste à enchaîner les cycles les uns après les autres.

fonctionnement mono-tâche périodique (synchrone)



Dans ce mode de fonctionnement, l'acquisition des entrées, le traitement du programme et la mise à jour des sorties s'effectue de façon périodique te ms selon un temps défini par configuration API .

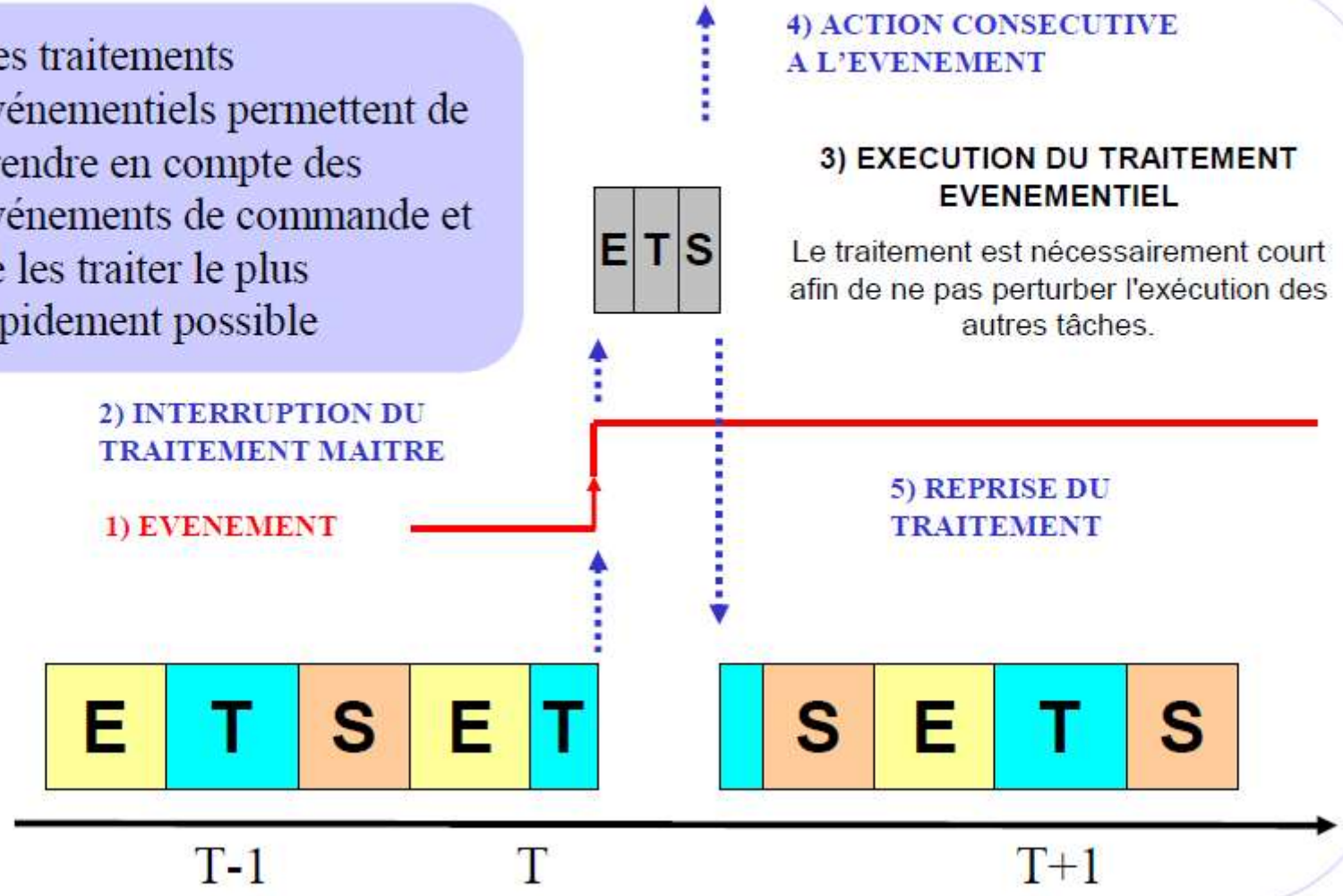
Retard dans le traitement de l'événement



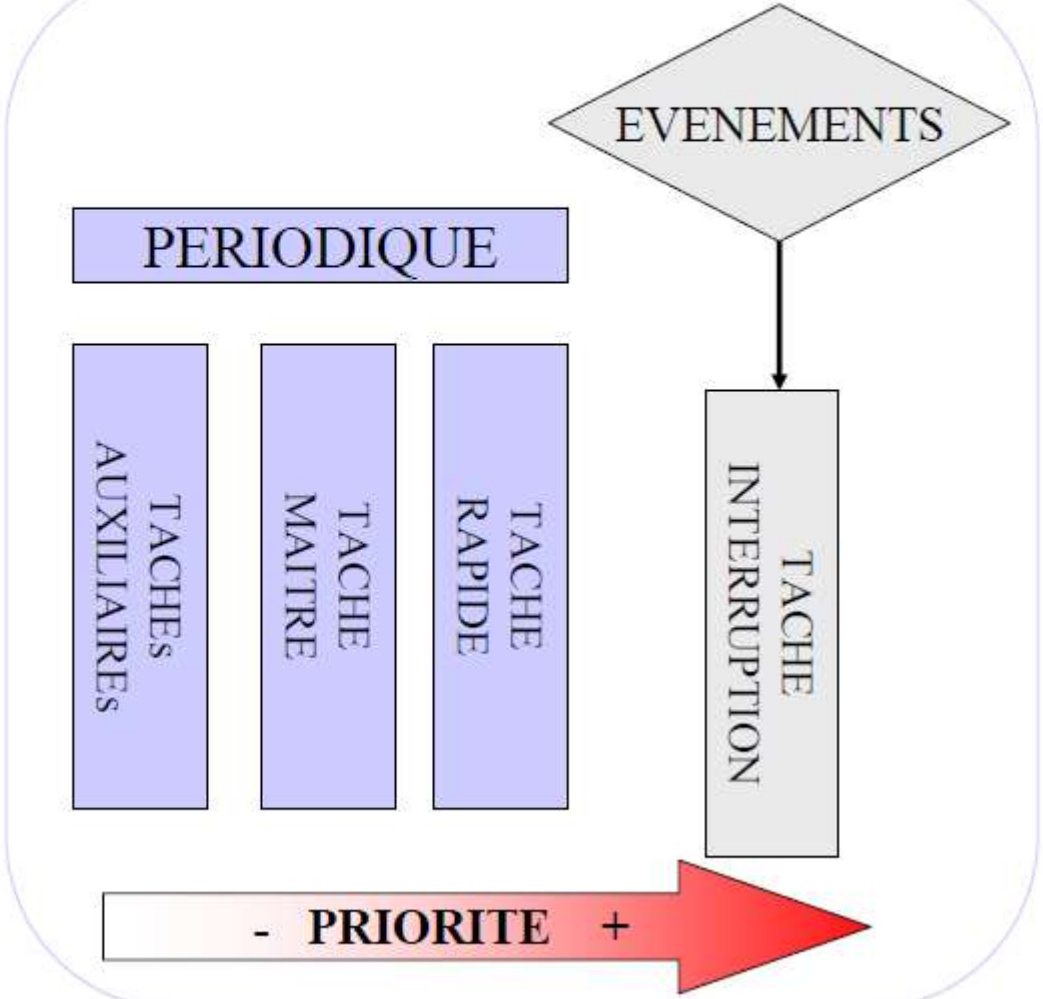
Les deux modes de traitements (cyclique ou périodique) sont appelé différé. Dans le pire des cas, il peut s 'écouler à peu prés 2 fois le temps de cycle moyen avant que l'UT réagisse à l'apparition d'un événement

Les interruptions

Les traitements événementiels permettent de prendre en compte des événements de commande et de les traiter le plus rapidement possible



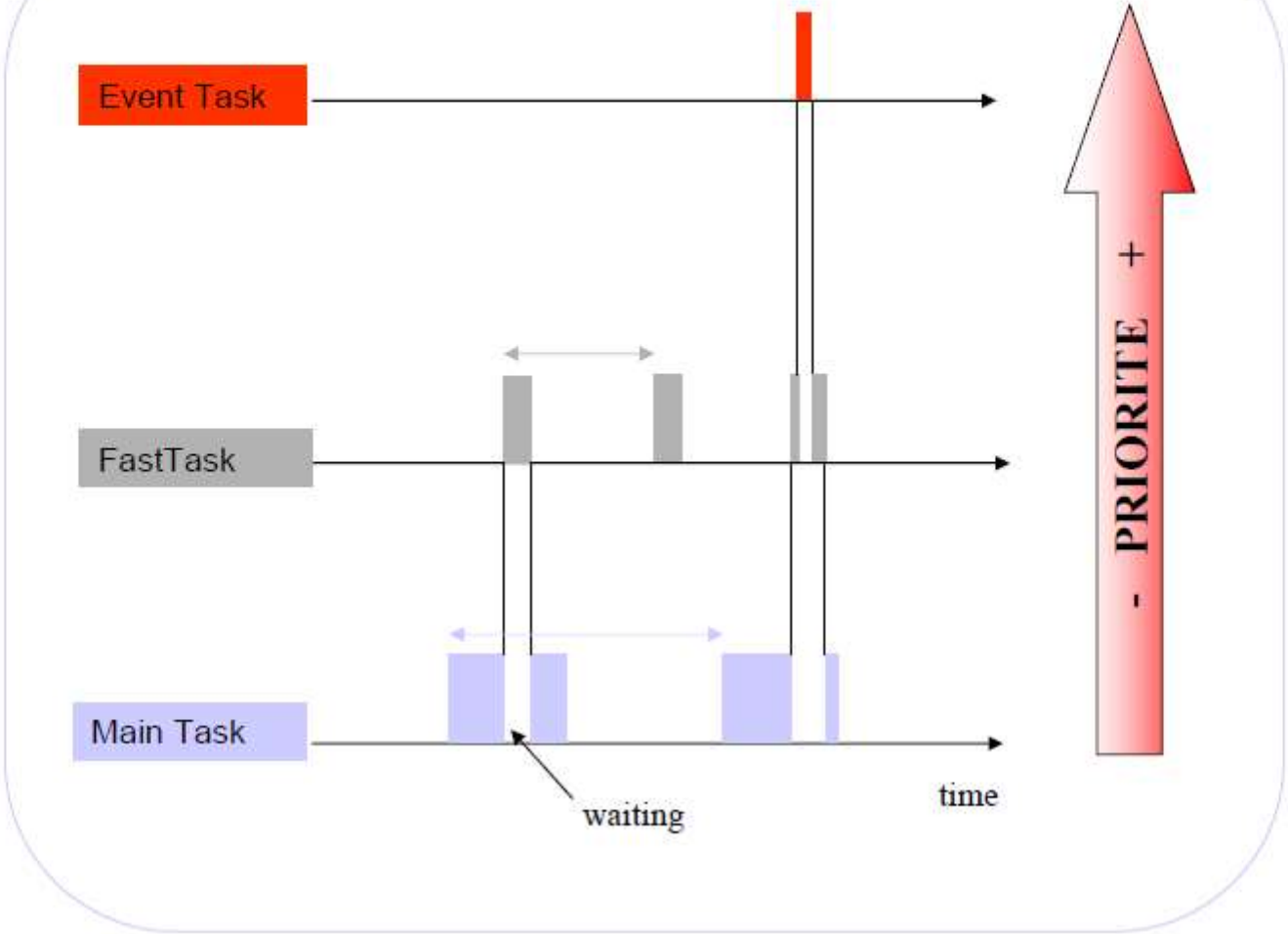
Traitement multitâches



La tâche rapide permet d'effectuer des traitements courts avec une priorité plus élevée que dans la tâche maître

Le traitement est nécessairement court afin de ne pas perturber l'exécution des autres tâches

Traitement multitâches



Inputs-Outputs

- I/O(Q)
- Can be directly on the PLC
- Can be on modules added to the PLC
- Can be on remote modules connected by an industrial network (ex: CAN)

Standardized variables

- Direct representation of variables
  - %NatureTypeAddress
    - Nature: I, Q, M
    - Type: X, B, W, D, L
    - Address
      - Simple: A number
      - Hierarchical, example: rack.module.channel
- %I0.1.0 Input as a bit (default value of Type), rack 0, module 1, channel 0
- Specific Types: %S => system, %K => constant

Variables

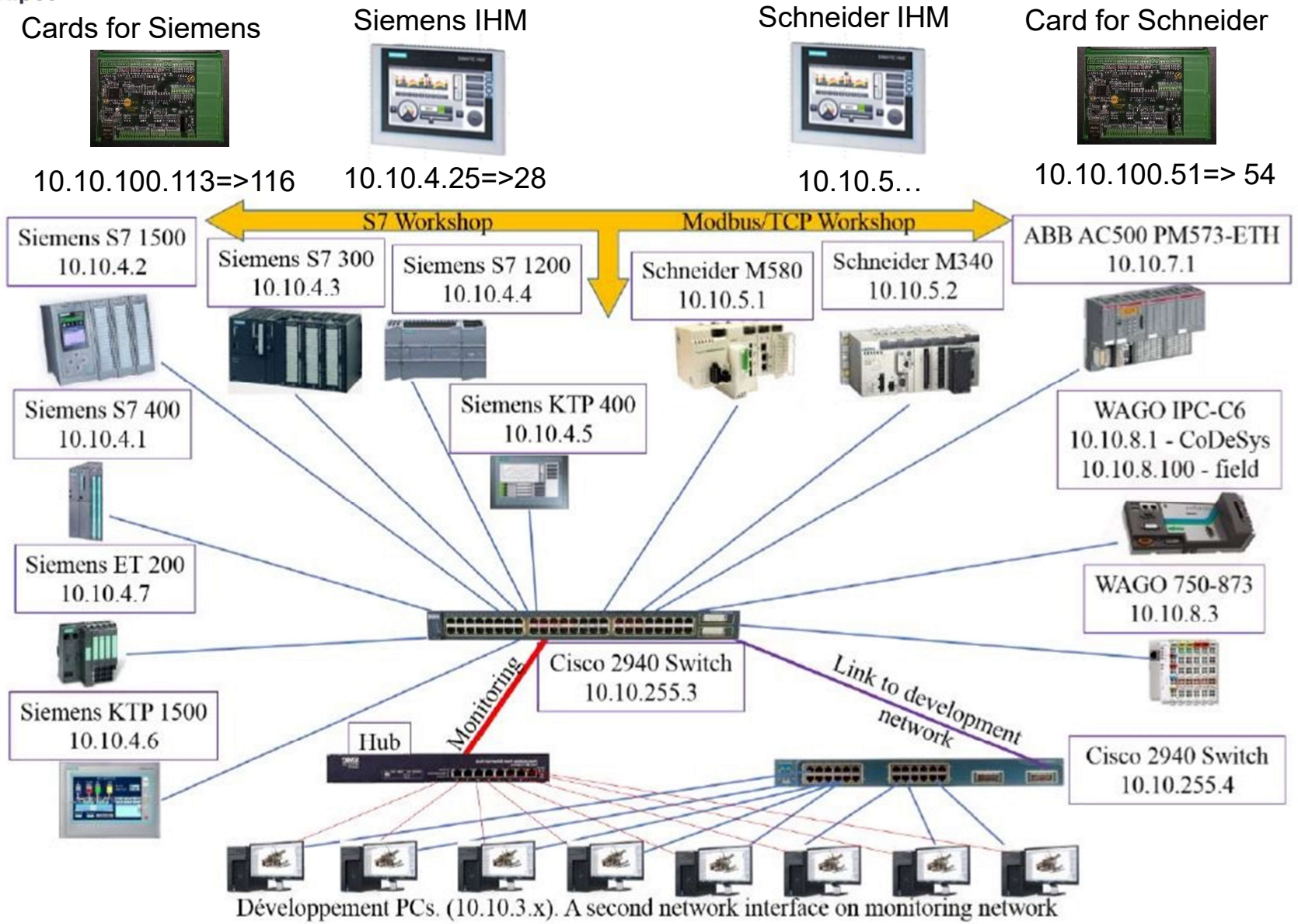
- Data types

Type	Désignation normalisée	Taille
Booleen	BOOL	1 bit
Entier signé	BYTE	8 bits
Entier signé	INT	16 bits
Entier signé	DINT	32 bits
Réels	REAL	32 bits
Temporisation	TIME	32 bits
Chaîne de caractères	STRING	255 caractères maxi

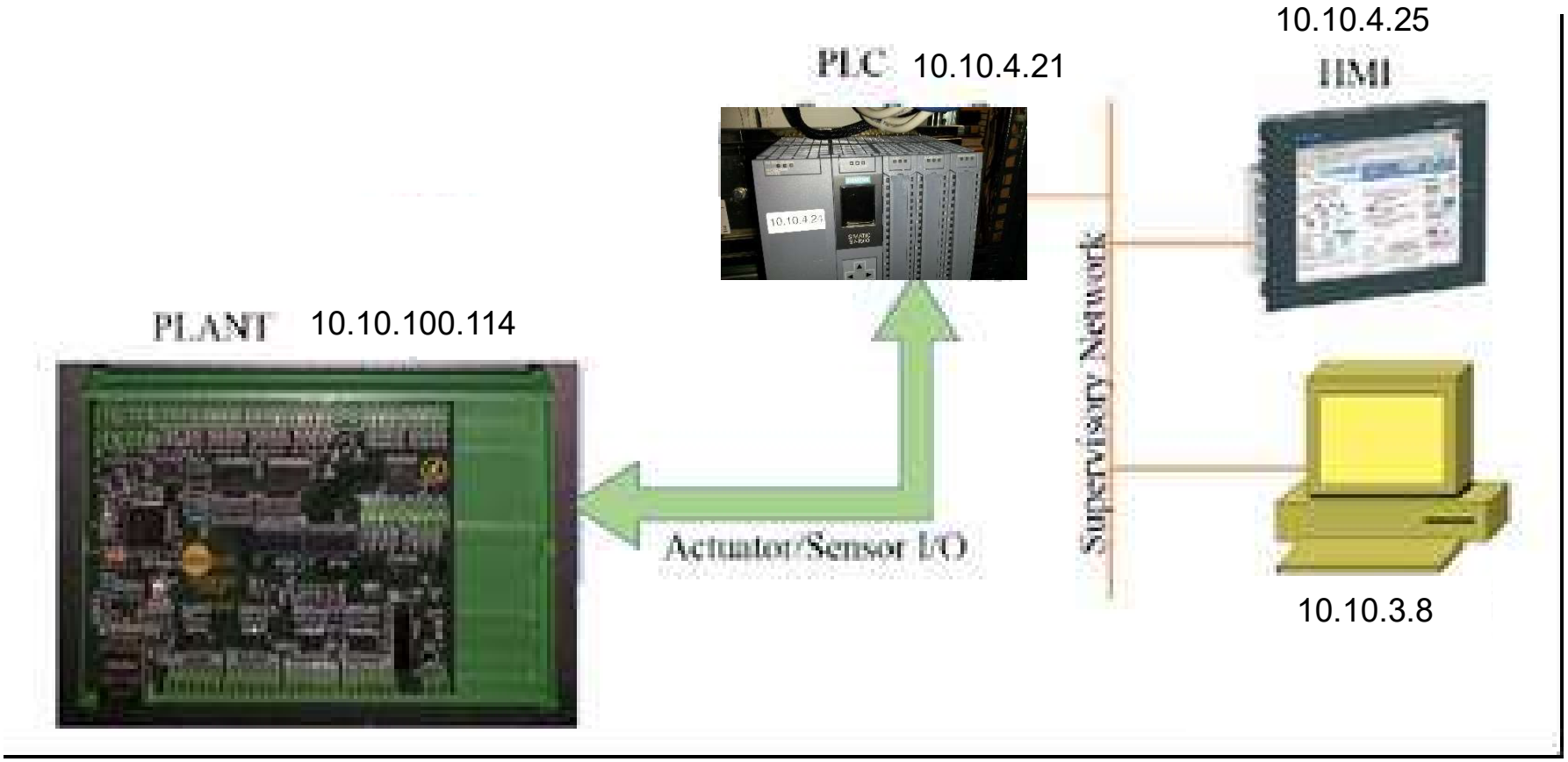
- A variable is represented by the % sign followed by a location, size and integer prefix separated by periods

Location prefix	Meaning	Size prefix	Meaning
I	Input location	None	1 bit
Q	Output location	X	1 bit
M	Memory location	B	1 byte (8 bits)
K	Constant location	W	1 word (16 bits)
		D	Double word (32 bits)
		L	Long word (64 bits)
		F	Floating word size (real)

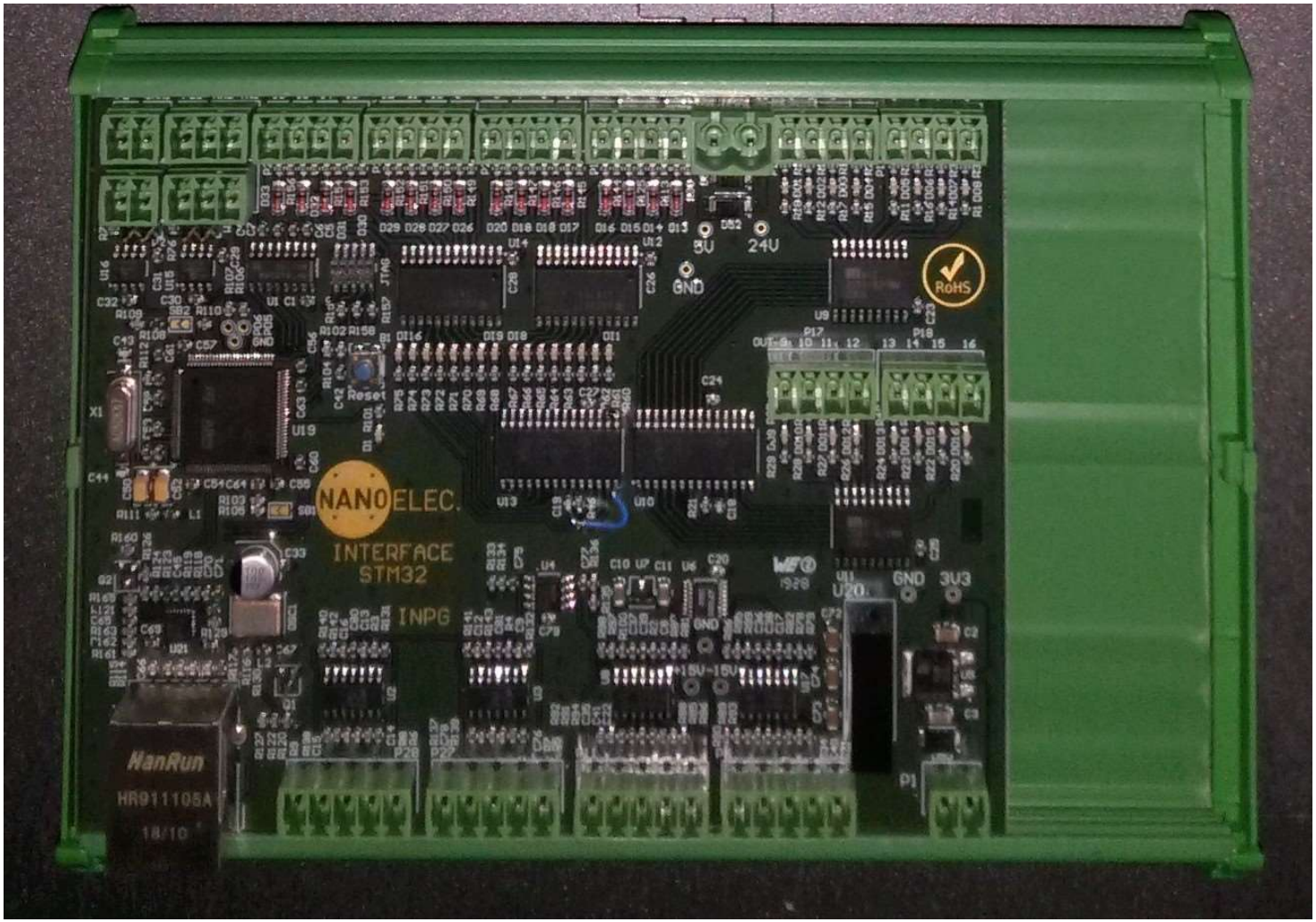
Lab Global architecture



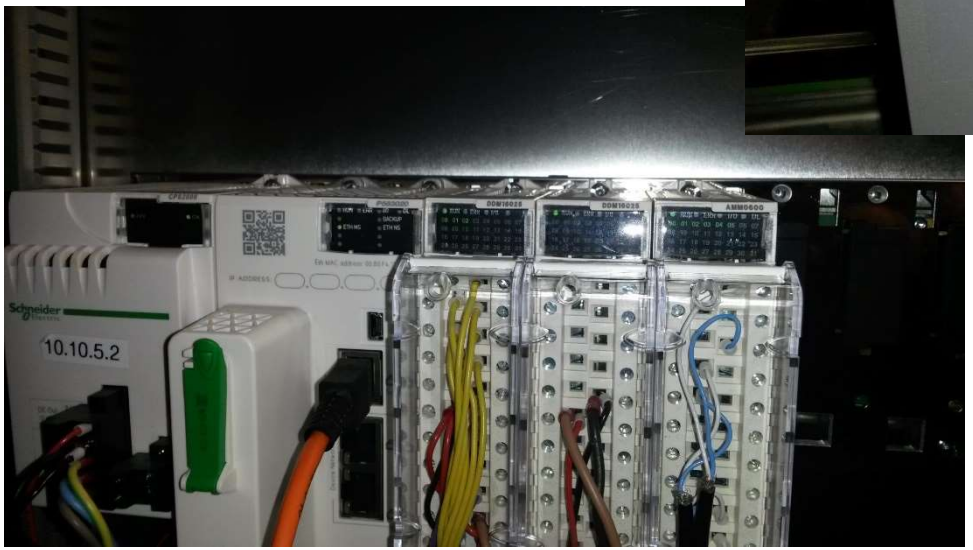
Siemens : 4 devices for the lab



Emulation/simulation card Hardware in The Loop




SIEMENS
10.10.4.24
SIMATIC
S7-1500

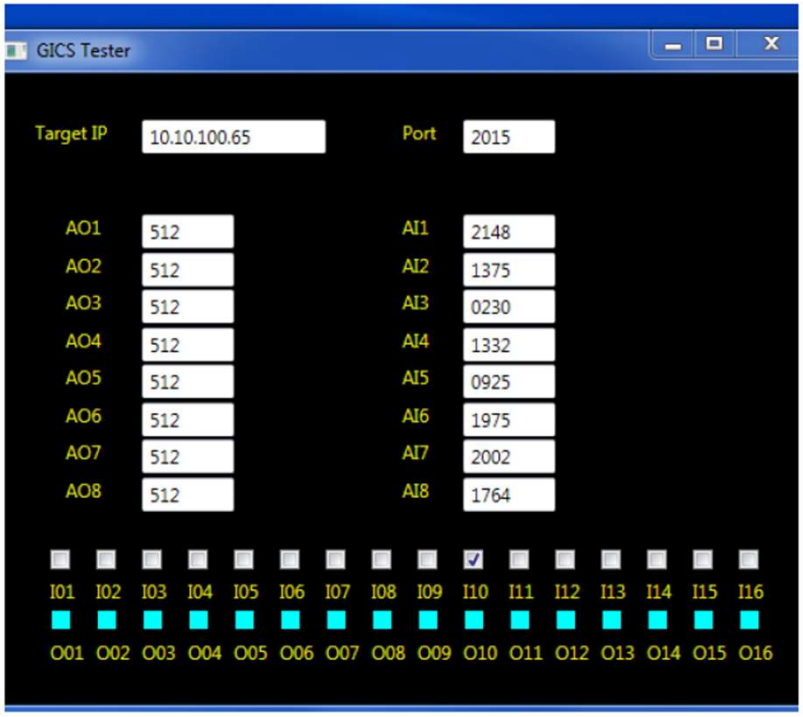

10.10.5.2

Correspondance PLC-card

Card	PLC	
10.10.100.114	10.10.4.21	Siemens
10.10.100.113	10.10.4.22	
10.10.100.115	10.10.4.23	
10.10.100.116	10.10.4.24	
10.10.100.51	10.10.5.61	Schneider
10.10.100.52	10.10.5.62	
10.10.100.53	10.10.5.63	
10.10.100.54	10.10.5.64	

GICS Tester

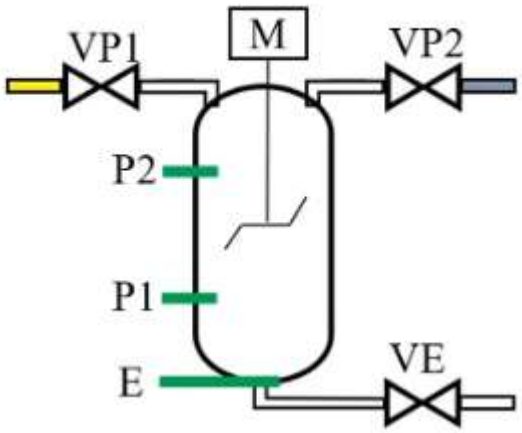
- I01 to I15 corresponds to Digital inputs (to be activated remotely)
- Q01 to Q15 corresponds to digital outputs (will be activated by the programme running on the PLC)
- A… corresponds to the Analog Inputs and Outputs



Example

Sequential Systems

- Behaviour characterised by a determinist automaton
- Outputs (actuators) as a function of the inputs (sensors) and states of the automaton
- Specifications in Grafcet and programmation in SFC (Sequential Function Chart)

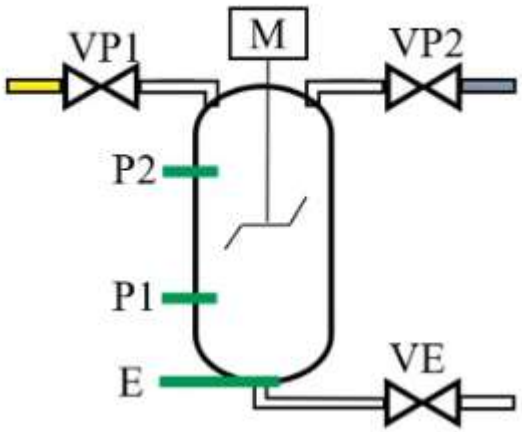


https://videos.univ-grenoble-alpes.fr/video/7273-tp-scada-05-grafcet/

https://videos.univ-grenoble-alpes.fr/video/7274-tp-scada-06-programmation-automate/

Example: control specification

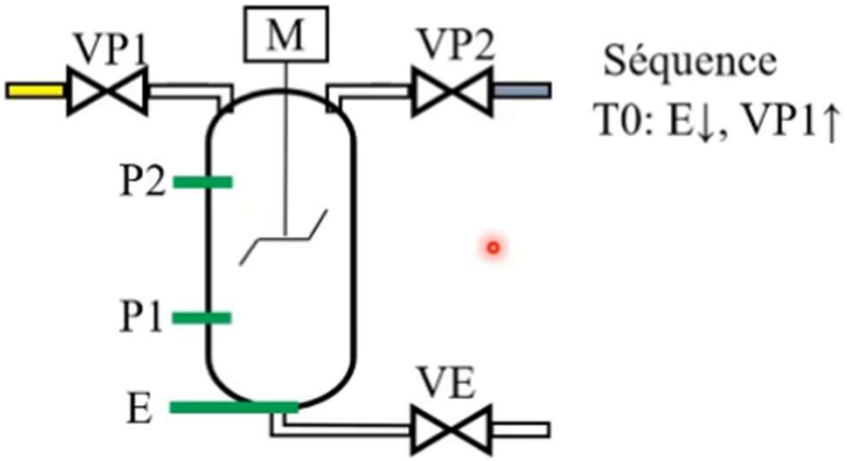
- Sequential behaviour: finite state machine, determitist automata, graphs with states and transitions
- Outputs are function of inputs and internal state of the automaton



- Process: mixture (blending) between two products in a tank containing a blender
- Mixture between Product 1 (yellow) and product 2 (blue)
- The blender is actuated by an engine (M)
- At the end the mixed product (green) will be evacuated through the valve VE

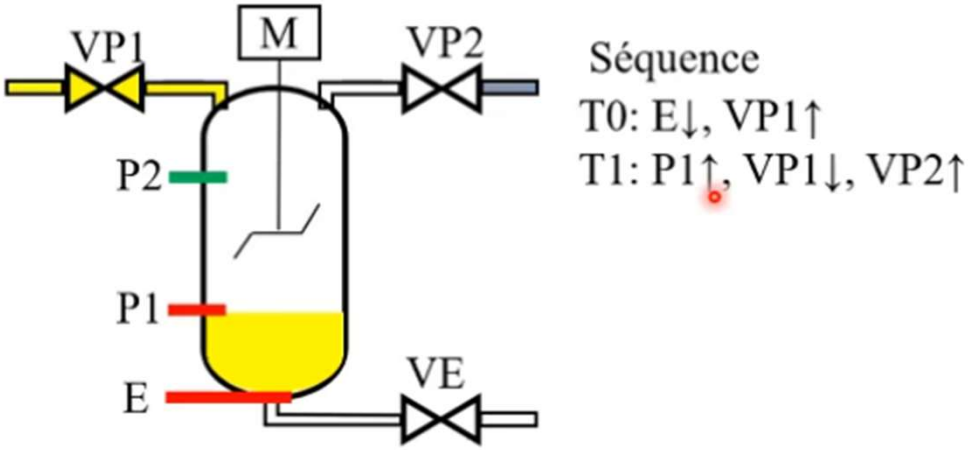
Example: control specification Transition 0

- Time T0
- When E switches from 1 to 0 (descending front), which means the tank is empty, this begins a new sequence



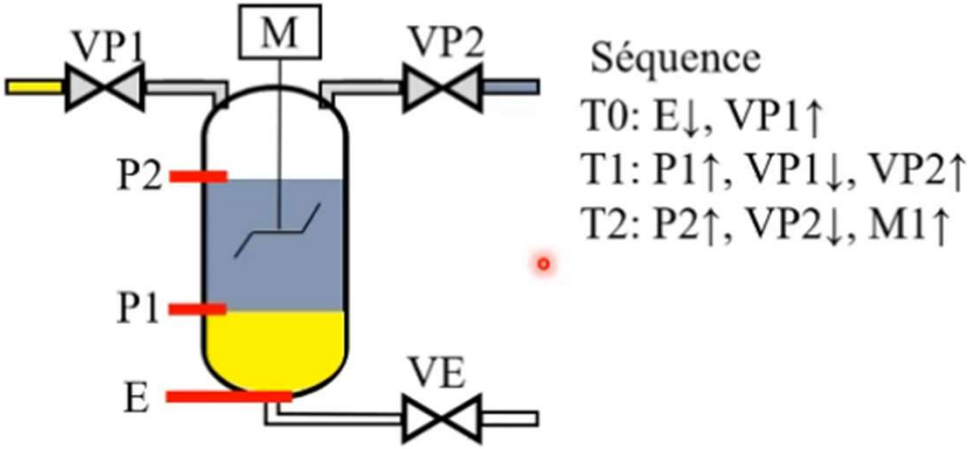
Example: control specification Transition 0 and 1

- Transition 0 : Opening of VP1 valve to add yellow product in the tank
- Once the P1 sensor (level) is reached => Transition 1: VP1 is closed, and VP2 (blue product) is open to add blue product



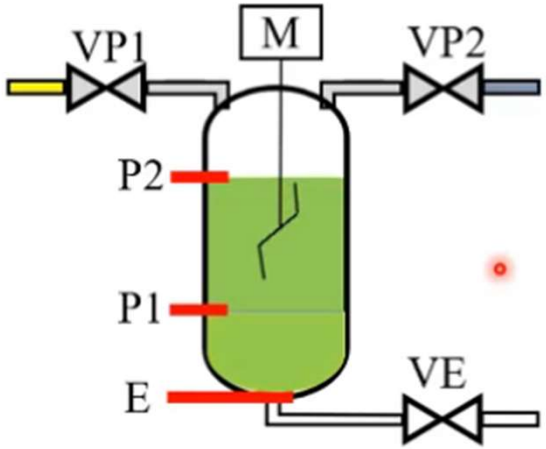
Example: control specification Transition 2

- Once the P2 sensor (level) is reached => Transition 2: VP2 is closed, and M (blender) is switches on



Example: control specification Transition 2

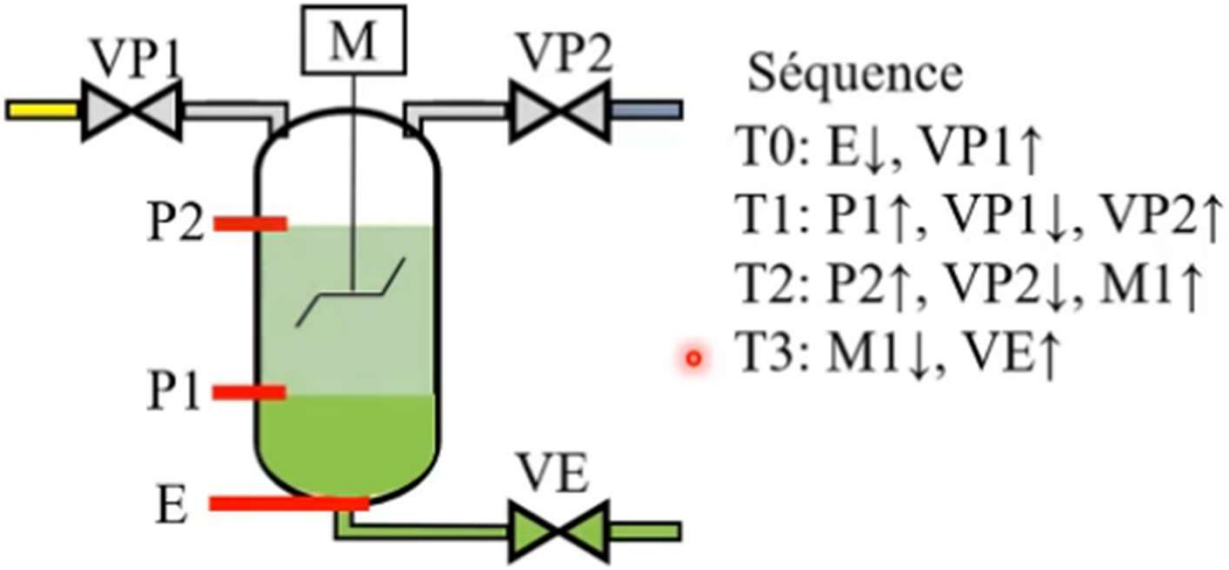
- The mixing (blending) operation will last a certain time. We use a timeout for that.



Séquence
T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑

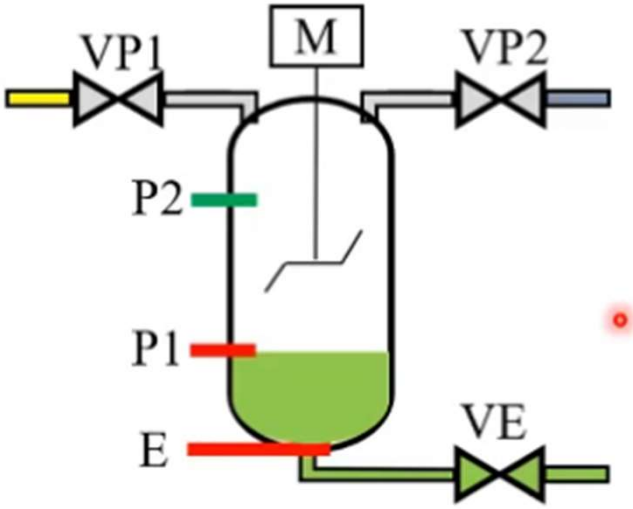
Example: control specification Transition 3

- At the end of the mixing, we stop the engine, and open VE, which is the evacuation valve.



Example: control specification Transition 4

- The tank is emtying, P2 is deactivated

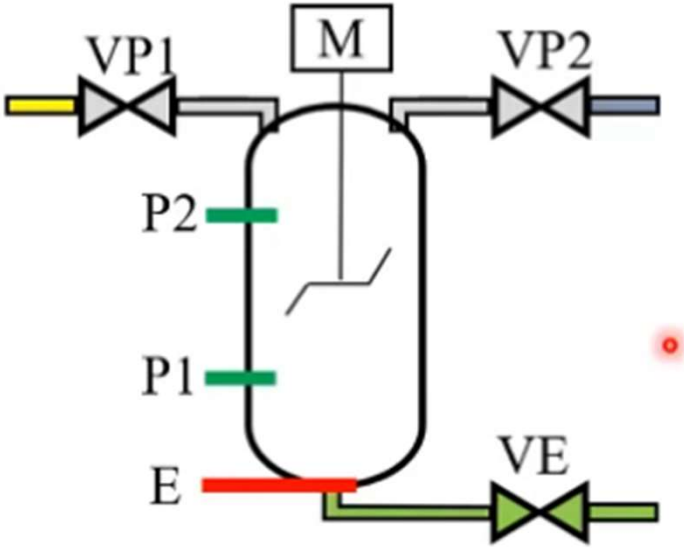


Séquence

T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑
T3: M1↓, VE↑
T4: P2↓

Example: control specification Transition 5

- Then P1 is deactivated

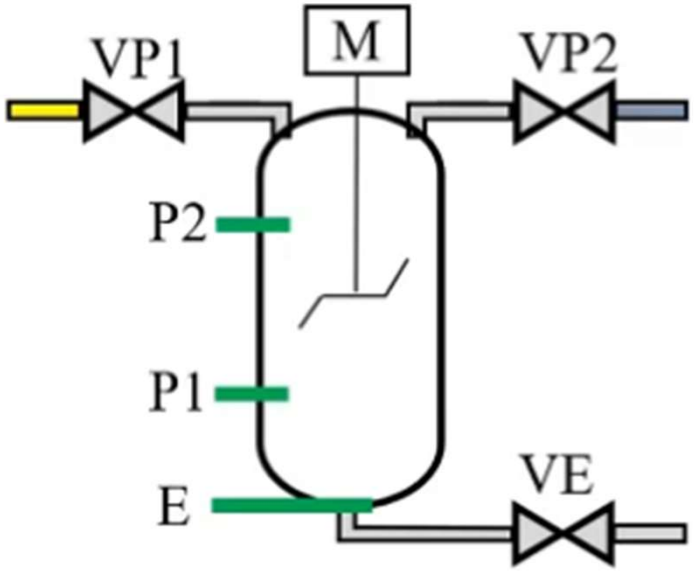


Séquence

T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑
T3: M1↓, VE↑
T4: P2↓
T5: P1↓

Example: control specification Transition 6

- VE is closed, the cycle is over
- We can begin again

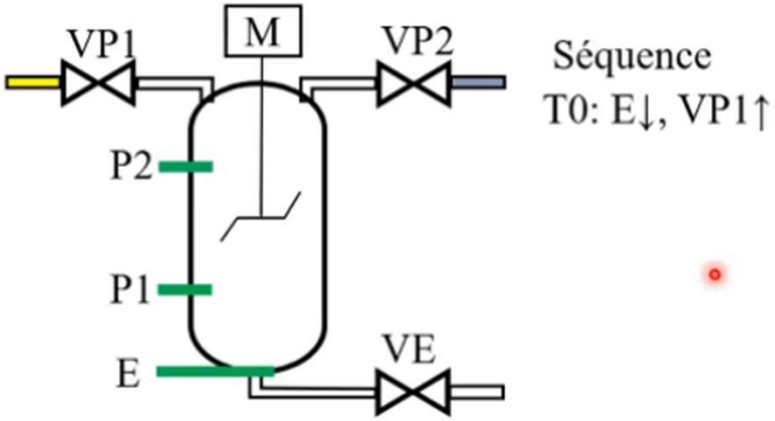


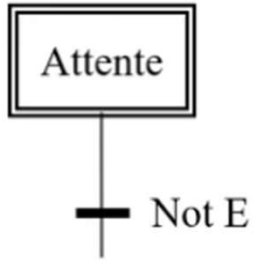
Séquence

T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑
T3: M1↓, VE↑
T4: P2↓
T5: P1↓
T6: VE↓

Equivalent Grafcet
Specification of the squential control

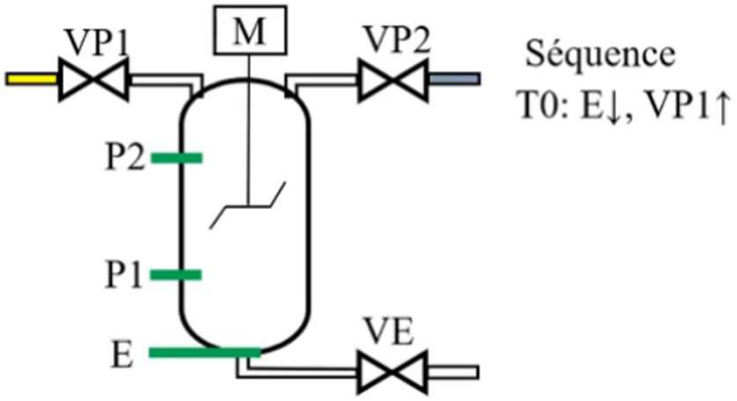
- Waiting
- The sequence when the sensor E is switched to 0

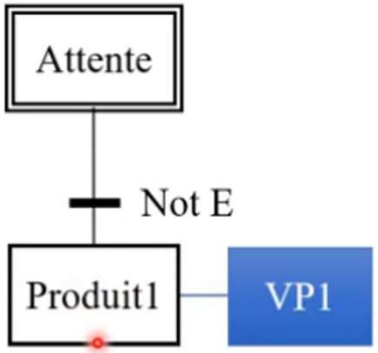




Grafcet

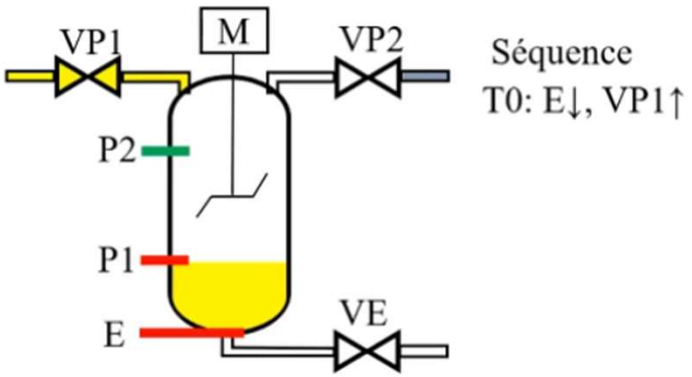
- State Produit1: VP1 is activated



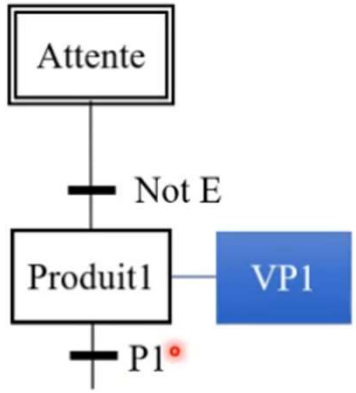


Grafcet

- When P1 is active (transition), the state Produit1 is over

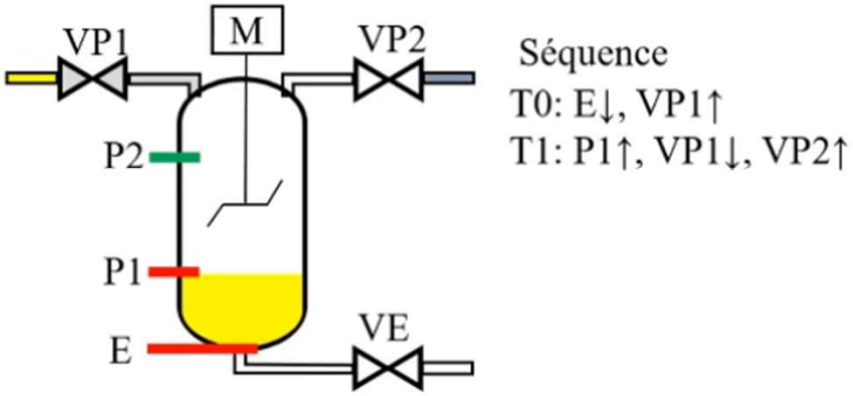


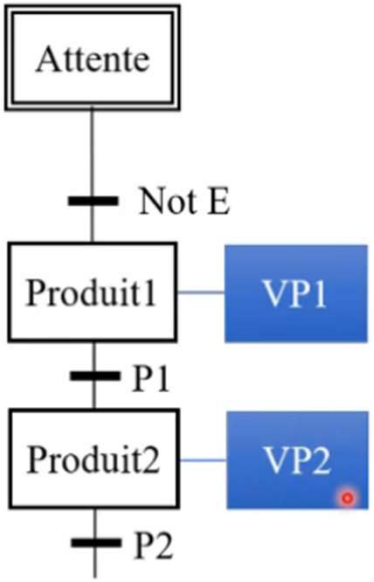
Séquence
T0: E↓, VP1↑



Grafcet

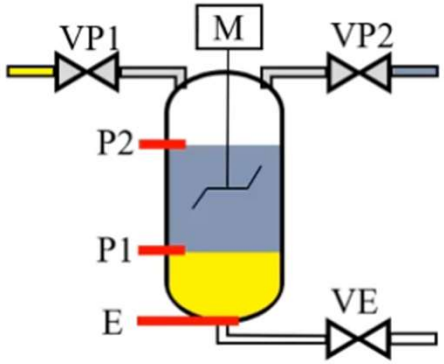
- We are now in state Produit2
- VP2 is active. VP1 is not active any more.
- NB: if an actuator is not activated explicitely, it is deactivated, it is called continuous action
- It exists also memorised actions



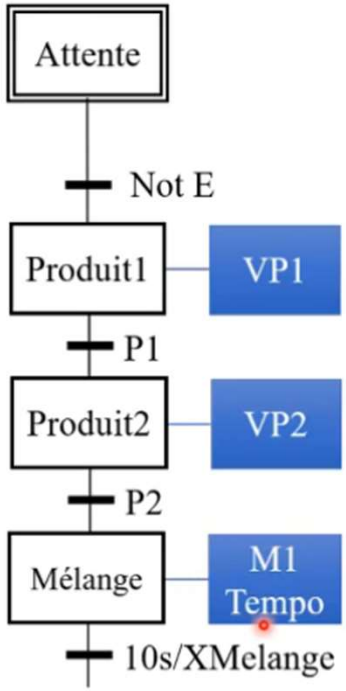


Grafcet

- When P2 is active => state Melange, M1 (engine is on) as well as a timeout (10s)
- After 10 s, the transition 10s/Xmelange is valid and …
- Xmelange means is related to the state Melange

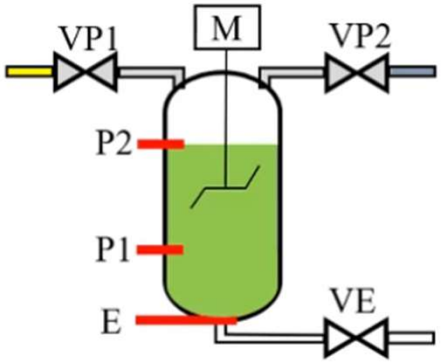


Séquence
T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑

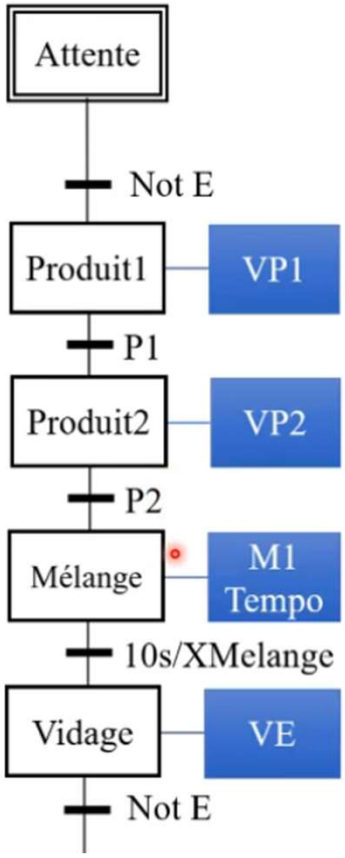


Grafcet

- … we reach the state Vidage, the valve VE is active
- Until Not E will be valid…

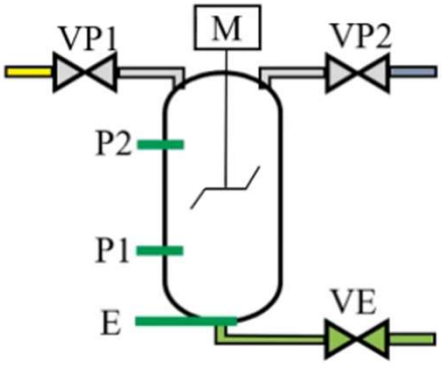


Séquence
T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑
T3: M1↓, VE↑

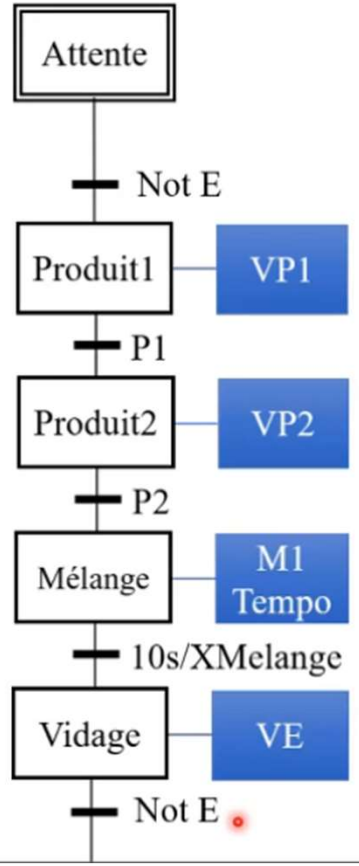


Grafcet

- Not E is valid (which means the tank is empty)
- The cycle is over…



Séquence
T0: E↓, VP1↑
T1: P1↑, VP1↓, VP2↑
T2: P2↑, VP2↓, M1↑
T3: M1↓, VE↑
T4: P2↓
T5: P1↓
T6: VE↓



Global methodology for labs

1. Define in TIA Portal/Unity the exact architecture, corresponding to the actual physical one, all the lots should be configured (it is equivalent to add the required libraries in IT)
2. Configure all the variables (I, O, M…)
3. Configure precisely the network interface (crucial step!), external clock for synchronization
4. Write a programme, the actions, possibly specific memories for Modbus frames
5. You can first work in simulation
    1. Choose the simulation
    2. Compile your programme (HW and SW)
    3. Send your programme to the PLC (here in simulation)
    4. Then run
6. You can then work with the real PLC
    1. Choose the actual PLC
    2. Compile your programme (HW and SW)
    3. Send your programme to the PLC (here on the actual PLC)
    4. Then run, you will need to interact using the simulation card
    5. You can also observe what's going on on the PLC using the debug mode
7. **Never forget to save from time to time what you do !**
8. **Always stop running and the debug mode when you want to send a new programme to the PLC**

References

- P_RAYMOND_BTS_MAI_Les_API
- Transmissions et réseaux, S. Lohier & D. Présent, Dunod, Paris, 2003.
- Cours Stéphane Mocanu, ENSE3, Industrial Communication Labs, 2016
- Cours Emmanuel Simeu, Polytech Grenoble, Supervision
- Cours de Blaise Conrard, Polytech Lille.
- Patrick Monassier, cours CESI 2009, Informatique industrielle.
- Pierre Bonnet, cours Université de Lille, Introduction à la supervision, 2010
- G. Boujat et P. Annaya, Automatique industrielle en 20 fiches, Dunod, 2007
- W. Bolton, Automates programmables industriels, Dunod, 2015.
- Certains compte-rendus d'étudiants

Université Grenoble Alpes

FACTORI 4.0 Erasmus +

Thank you !

Pour aller plus loin

- Sauts, sauts impriqués : p.222, API Bolton
- Sous-programmes p. 226
- Temporisateurs à enclenchement et à déclenchement (chap 9)
- Compteurs et séquenceurs (chap 10)
- …

- Voir le livre LE GRAFCET : de nouveaux concepts => très bien pour faire un cours introductif

Gestion des données

- Temporisateurs, compteurs, relais internes, sont impliqués dans une gestion de bits individuels.
- Les API disposent d'opérations qui concernent des blocs de données représentant une valeur : ces blocs sont appelés mots
- Un bloc de données est indispensable si l'on veut représenter des nombres à la place d'une simple entrée Tout Ou Rien
- La gestion des données comprend :
  - Opérations qui déplacent ou transfèrent des informations numériques depuis un mot de mémoire vers un autre enregistré à un emplacement différent
  - Pour permettre la comparaison de valeurs ou des opérations arithmétiques simples
  - Ex : comparer une valeur numérique à une valeur de référence pour déclencher une action si la valeur actuelle est inférieure à la valeur fixée

Management of data

- Timers, counters, internal relays, are involved in a management of individual bits.
- PLCs have operations that concern data blocks representing a value: these blocks are called words
- A data block is essential if you want to represent numbers instead of a simple On/Off input
- Data management includes:
  - Operations that move or transfer numerical information from one memory word to another stored in a different location
  - To allow comparison of values or simple arithmetic operations
  - E.g.: compare a numeric value to a reference value to trigger an action if the current value is less than the set value

– Constitution des blocs fonctionnels FBD

Un FBD est constitué :

– d'un nom ;

– d'un commentaire ;

– de paramètres :

- **d'entrées :** ce sont les données à fournir au FBD par le programme application. Ces paramètres en lecture seule ne peuvent pas être modifiés dans le code du FBD.
- **de sorties :** ce sont les données élaborées par le FBD à destination du programme application (paramètres en écriture).
- **d'entrées/sorties :** ce sont des paramètres d'entrées modifiables dans le code du FBD (paramètres en lecture/écriture).

– de variables :

- **Variables globales :** variables internes utilisées dans le traitement et accessibles par l'utilisateur ou par le programme application en dehors du code FBD.
- **Variables locales :** variables internes au code du bloc fonction, ces variables sont calculées et exploitées à l'intérieur même du FBD mais n'ont aucun lien avec l'extérieur du FBD. Ces variables sont utiles pour la programmation du bloc mais n'ont pas d'intérêt pour l'utilisateur du bloc (par exemple résultat d'un calcul intermédiaire...).

– d'un code.

Le code d'un FBD est son programme interne définissant les traitements à effectuer en fonction des paramètres déclarés. Le code peut être écrit dans les langages IL, ST ou LD.

IHM Siemens sur la plate-forme GICS

IHM TP 700 Comfort 6AV2 124-0GC01-0AX0