

TP 1 : Automatisme

Exercice 0 : Lecture et paramétrage

Pour faire ce TP, nous avons utilisé le logiciel Simatic step 7 qui permet d'accéder aux automates Siemens dans le but de les programmer.

Avant de commencer l'exercice 1 du TP, nous sommes passés par plusieurs étapes :

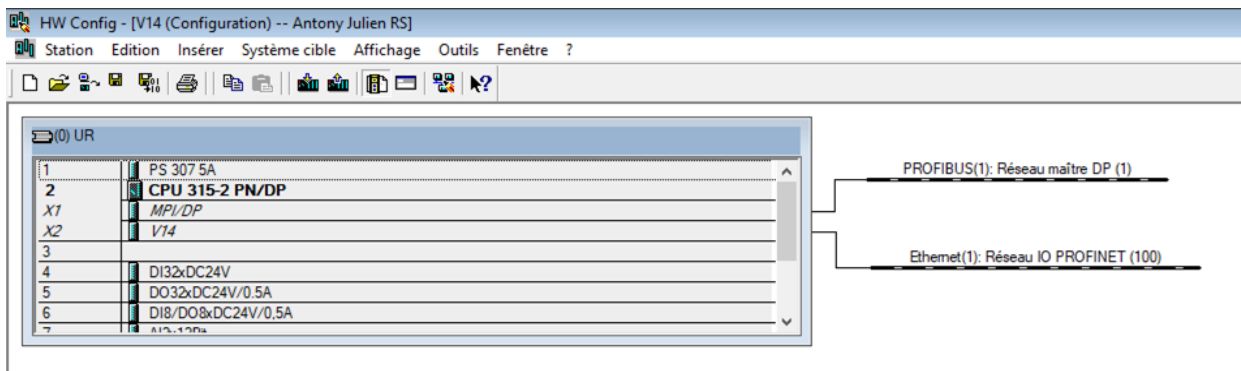
Étape 1 : Créer le projet

Un projet contient la description complète de notre automatisme.

Il comporte deux grandes parties : la description du matériel, et la description du fonctionnement (les programmes que nous allons créer par la suite dans les exercices).

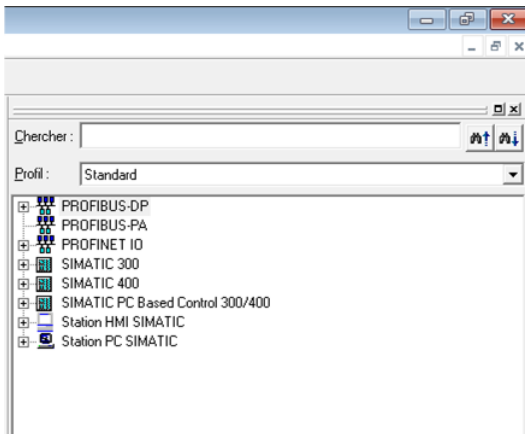
Étape 2 : Décrire le matériel grâce à la documentation sur step 7

On a défini chaque composant 1 à 1 ainsi que les deux connexions différentes (Ethernet et Profibus) en faisant en sorte de bien vérifier le nom des modules, leurs références et leurs adresses car parfois les composants ont le même nom mais des adresses et des références différentes.



Emplacement	Module	Référence	Firmware	Adresse MPI	Adresse d'entrée	Adresse de sortie	Commentaire
1	PS 307 5A	6ES7 307-1EA00-0AA0					
2	CPU 315-2 PN/DP	6ES7 315-2EG10-0AB0	V2.3				
X1	MPI/DP				2047		
X2	V14				2046		
3							
4	DI32xDC24V	6ES7 321-1BL00-0AA0			0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0				0...3	
6	DI8/DO8xDC24V/0.5A	6ES7 323-1BH01-0AA0			4	4	
7	AI2x12Bit	6ES7 331-7KB02-0AB0			5...9		
8							
9							
10							
11							

Attention : il faut bien paramétrer les entrées et sorties de chaque élément en suivant le document sinon on aura des problèmes par la suite dans les exercices.



À droite de notre page, cette fenêtre nous permet de voir la liste de tous les composants et d'effectuer une recherche dans tous ceux-ci.

Il faut veiller à bien enregistrer la description de notre automate dans un projet dans lequel ne figure aucun programme.

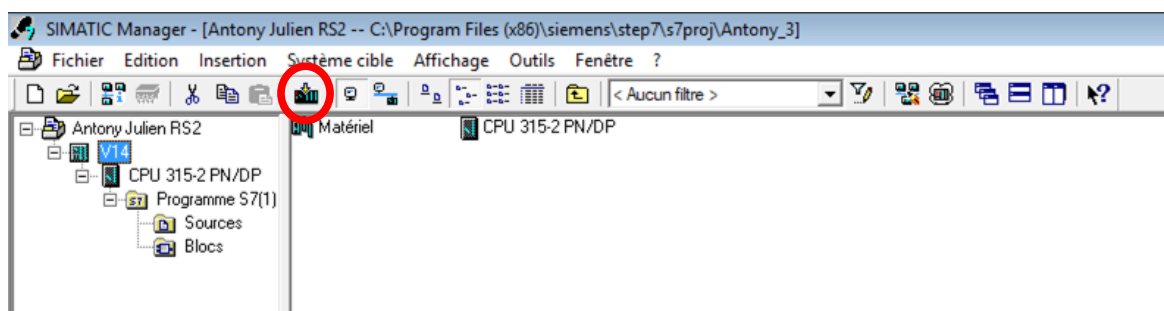
Nous avons ainsi commencé tous nos exercices à partir du même projet et donc du même matériel.


Néanmoins, à chaque exercice, nous avons enregistré nos programmes sous un nom différent pour toujours garder le projet initial qui ne sera jamais modifié.

Après avoir fait nos programmes, il serait bon de vérifier qu'ils fonctionnent.

Pour cela, il faut pouvoir charger le programme sur l'automate.

Comment lancer le programme sur l'automate ?




Pour tester le programme on clique d'abord sur la valise V14 (notre valise) puis on appuie sur l'icône  ce qui permet de charger le programme sur l'automate.

Attention : Il faut ouvrir un seul projet à la fois sinon l'automate peut ne plus fonctionner (nous avons vu une LED s'allumer en rouge après avoir ouvert deux projets différents).

On peut ensuite tester notre programme sur l'automate en éteignant et allumant les boutons et vérifier le comportement des sorties en fonctions des entrées.

On peut aussi visualiser le résultat du programme directement sur l'ordinateur.

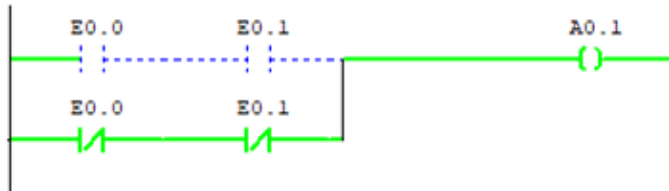
Pour cela il suffit de sélectionner dans la barre d'outils l'icône 

On obtient l'affichage suivant :

OB1 : "Main Program Sweep (Cycle)"

Commentaire :

▣ Réseau 1: combinatoire simple



Exercice 1 : Combinatoire simple : une sortie

Pour étudier cet exercice, on a, dans un premier temps, détaillé toutes les possibilités.

On définit e1, e2 et e3 les 3 entrées.

1 correspond à une entrée allumée (on appuie dessus) et 0 une entrée éteinte (on n'appuie pas dessus).

Le but de ce programme est que la sortie s'allume (LED) quand la somme des numéros d'entrée est paire.

e1	e2	e3	Résultat
0	0	0	Pair
0	0	1	Impair
0	1	0	Pair
1	0	0	Impair
1	1	0	Impair
0	1	1	Impair
1	0	1	Pair
1	1	1	Pair

→ Rien n'est allumé (0+0+0=0)

→ L'entrée e3 est allumée, les autres éteintes

→ L'entrée e2 est allumée, les autres éteintes (0+2+0=2)

→ L'entrée e1 est allumée, les autres éteintes

→ Les entrées e1 et e2 sont allumées simultanément

→ Les entrées e2 et e3 sont allumées simultanément

→ Les entrées e1 et e3 sont allumées simultanément (1+0+3=4)

→ Les 3 entrées sont allumées simultanément (1+2+3=6)

Analysons ces résultats avec un tableau de Karnaugh dans le but d'obtenir une équation.

		e1e2			
		00	01	11	10
e3	0	1	1	0	0
	1	0	0	1	1

On obtient l'équation suivante : $S = e1 \cdot e3 + \overline{e1} \cdot \overline{e3}$

Remarque : La théorie nous dit que l'entrée e2 n'entre pas dans l'équation, elle ne sera donc pas considérée dans le programme suivant.

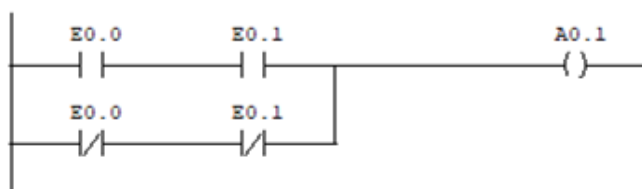
Nous aurons uniquement deux boutons.

Attention : Il faut veiller ici à bien utiliser le langage CONT comme précisé dans le TP.

OB1 : "Main Program Sweep (Cycle)"

Commentaire :

▣ Réseau 1: combinatoire simple



E0.0 correspond à l'entrée e1 et au bouton 1 sur l'automate, E0.1 correspond à l'entrée e3 et au bouton 3 sur l'automate et A0.1 est la sortie. (Voir détails en annexe)

Après avoir créé le programme, nous aurions pu définir des mnémoniques.

Les mnémoniques permettent de simplifier la compréhension du programme en définissant chaque entrée et sortie.

Dans le prochain exercice, vous pourrez voir un exemple d'utilisation des mnémoniques.

On a chargé notre programme sur l'automate grâce à la démarche détaillée dans l'exercice 0.
On l'a ensuite testé manuellement sur l'automate en actionnant les boutons et en considérant toutes les possibilités :

Quand on actionne 1 on n'a rien.

Quand on actionne 3 rien non plus.

Quand on n'actionne aucun des boutons la sortie s'allume (0 est un nb pair).

Quand on actionne les deux boutons la sortie s'allume ($3+1=4$ nb pair).

Comme e_2 n'intervient pas dans l'équation nous ne considérons pas les possibilités $0+2+0$ et $1+2+3$.

Notre programme est donc **correct**.

Exercice 2 : Combinatoire

On va étudier le fonctionnement d'un tapis amenant des pièces vers deux machines.
 Dans un premier temps, on cherche les équations de PG, PD et E en fonction de g1, g2, d1 et d2.

On définit 4 entrées (on considère au maximum deux pièces de chaque côté) :

g1 = 1^{ère} pièce à gauche

g2 = 2^{ème} pièce à gauche

d1 = 1^{ère} pièce à droite

d2 = 2^{ème} pièce à droite

On définit ensuite les 3 sorties :

PD = il y a plus de pièces en attente à droite qu'à gauche

PG = il y a plus de pièces en attente à gauche qu'à droite

E = égalité (il y a autant de pièces à droite qu'à gauche)

· ANALYSE DES POSSIBILITÉS

On a 3 tableaux de Karnaugh (voir le détail des calculs en annexe) :

Pour PD :

		g1g2			
		00	01	11	10
d1d2	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	0	0	0

$$PD = d1 \cdot d2 \cdot (\overline{g1} + \overline{g2}) + \overline{g1} \cdot \overline{g2} \cdot (d1 + d2)$$

Pour PG :

		g1g2			
		00	01	11	10
d1d2	00	0	1	1	1
	01	0	0	1	0
	11	0	0	0	0
	10	0	0	1	0

$$PG = \overline{d1} \cdot \overline{d2} \cdot (g1 + g2) + g1 \cdot g2 \cdot (\overline{d1} + \overline{d2})$$

Pour E :

		g1g2			
		00	01	11	10
d1d2	00	1	0	0	0
	01	0	1	0	1
	11	0	0	1	0
	10	0	1	0	1

$$E = \overline{PG} \cdot \overline{PD}$$

On a essayé de calculer E en fonction des entrées mais nous n'y sommes pas arrivés.

On ne peut en effet rien regrouper dans ce tableau de Karnaugh ce qui rend l'équation très longue.

On peut cependant comprendre l'équation de la manière suivante : quand on a une égalité, on n'a ni plus de pièces à droite, ni plus de pièces à gauche.

Dans un second temps, on va essayer de trouver les équations liant les sorties AD et AG en fonction des entrées (pp, PG, PD, E, g, d).

Ces équations sont celles qui orientent l'aiguillage.

On définit 6 entrées :

PD = il y a plus de pièces en attente à droite qu'à gauche

PG = il y a plus de pièces en attente à gauche qu'à droite

E = égalité (il y a autant de pièces à droite qu'à gauche)

pp = détection de l'arrivée d'une pièce via le capteur

g = position de l'aiguillage à gauche

d = position de l'aiguillage à droite

On définit les deux sorties :

AD = aller à droite si :

- le capteur pp détecte une pièce et il y a plus de pièces à gauche

- ou on a égalité mais l'aiguillage est positionné à gauche au moment où pp détecte la pièce

AG = aller à gauche si :

- le capteur pp détecte une pièce et il y a plus de pièces à droite

- ou on a égalité mais l'aiguillage est positionné à droite au moment où pp détecte la pièce

On obtient les équations suivantes :

$$AD = pp \cdot (PG + E \cdot g)$$

et

$$AG = pp \cdot (PD + E \cdot d)$$

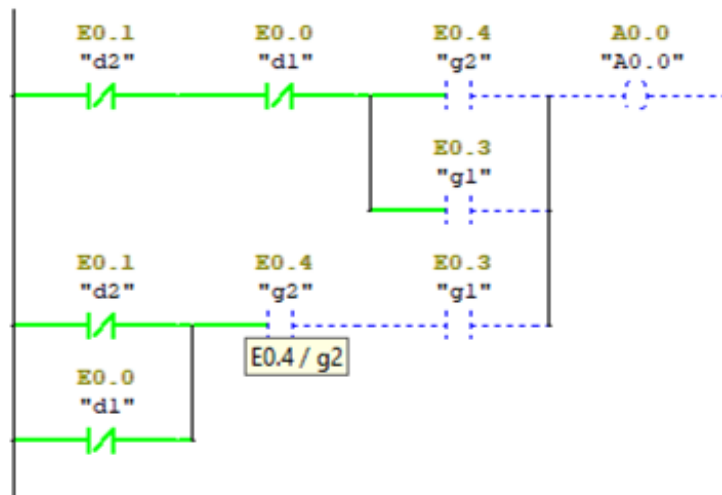
- PROGRAMME en langage CONT

PG :

OB1 : "Main Program Sweep (Cycle)"

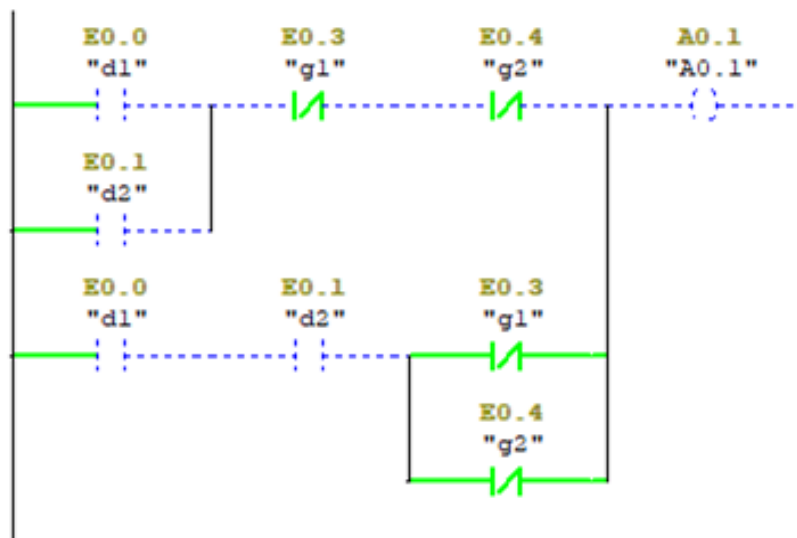
d2=E0.1 / d1=E0.0 / g1=E0.3 / g2=E0.4 A0.0= Plus a gauche A0.1 = Plus à droite

▣ Réseau 1 : Pour G

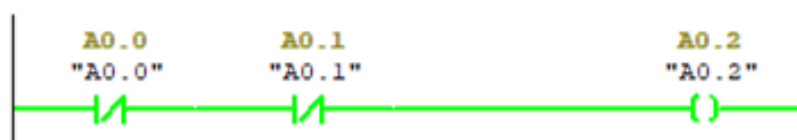


PD :

▣ Réseau 2 : Pour D



▣ Réseau 3 : Egalité



On a ici défini les mnémoniques de la manière suivante :

Programme S7(1) (Mnémoniques) -- Antony Julien RS2\V14\CPU 315-2 PN/DP					
	Etat	Mnémonique /	Opérande	Type de do	Commentaire
1		A0.0	A 0.0	BOOL	
2		A0.1	A 0.1	BOOL	
3		A0.2	A 0.2	BOOL	
4		d1	E 0.0	BOOL	
5		d2	E 0.1	BOOL	
6		g1	E 0.3	BOOL	
7		g2	E 0.4	BOOL	
8					

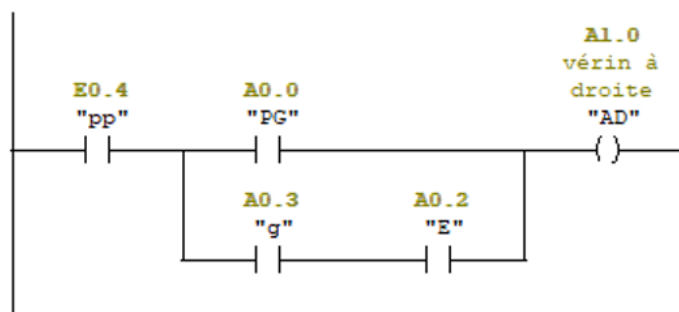
Comme dans l'exercice précédent on a chargé le programme avec, l'ordinateur et testé manuellement en actionnant les boutons. Nos programmes semblent corrects.



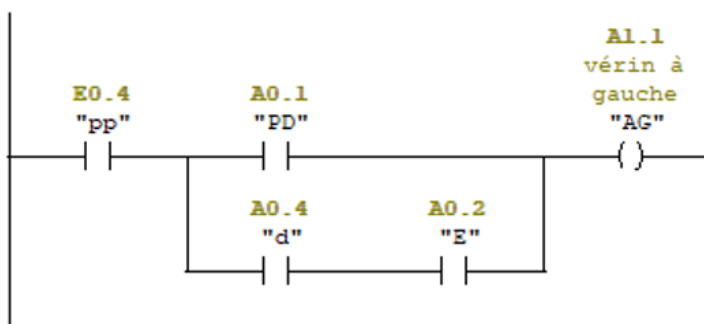
on a visualisé sur

On a ensuite programmé AD et AG sans les tester par manque de temps.

☐ Réseau 4 : AD



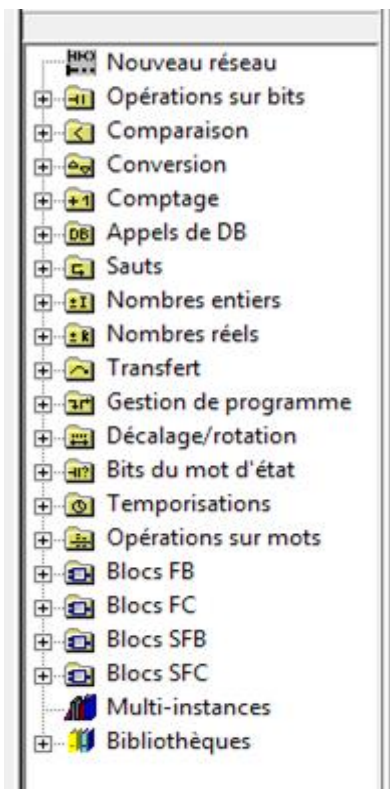
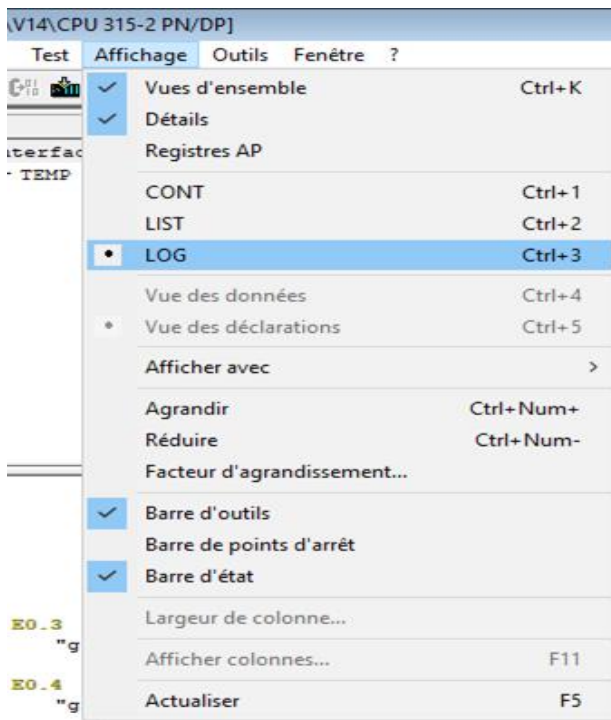
☐ Réseau 5 : AD



Exercice 3 : Langage en LOG

Pour visualiser le programme sous un autre langage, c'est très simple il faut se rendre dans la section affichage et sélectionner le langage souhaité.

(Il ne faut pas refaire entièrement le programme dans un autre langage comme nous avons commencé à le faire avant de voir cette option.)



On intègre chaque élément du programme (en les glissant dans la fenêtre dédiée à la programmation) via cette liste des composantes.

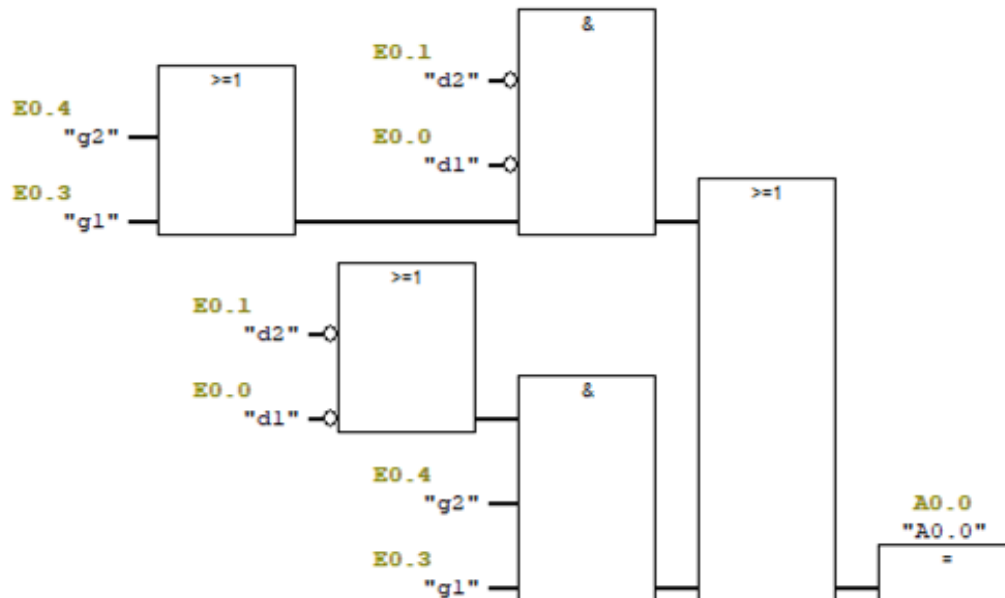
La conversion en LOG nous donne les programmes suivants :

Pour PG :

OB1 : "Main Program Sweep (Cycle)"

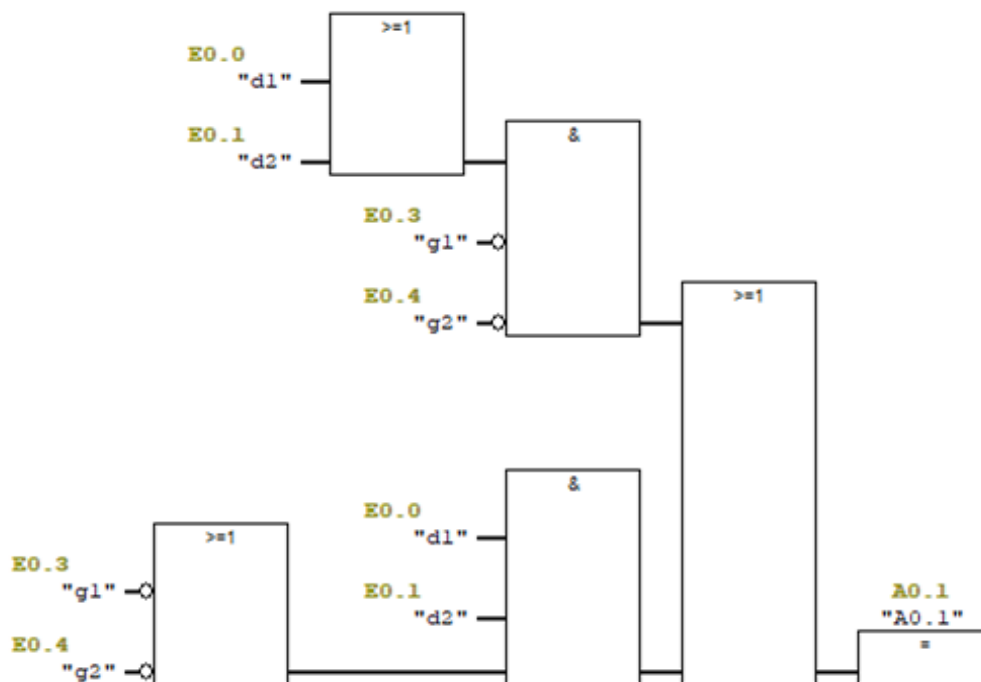
d2=E0.1 / d1=E0.0 / g1=E0.3 / g2=E0.4 A0.0= Plus a gauche A0.1 = Plus à droite

▣ Réseau 1: Pour G



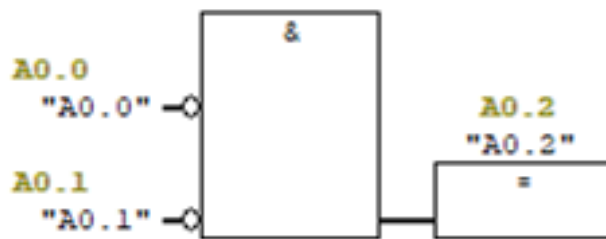
Pour PD :

▣ Réseau 2: Pour D



Pour Égalité :

▣ Réseau 3 : Egalité



Le langage en LOG (opérateurs logiques) est celui que l'on utilise en cours.

On le connaît déjà, il n'y a donc pas eu de phase d'apprentissage et de compréhension de ce langage.

Le programme en LOG nous semble plus facile à comprendre car plus visuel, on voit directement les opérateurs agissant sur les entrées.

Néanmoins, les répétitions dans les entrées (deux fois g1, g2, d1 et d2 pour PG et PD) rendent le programme vraiment imposant à l'affichage.

Le langage CONT est, lui, plus compact et compréhensible plutôt rapidement.

Exercice 4 : Langage en LIST

Comme précédemment, on sélectionne dans l'affichage le langage LIST.

OBI : "Main Program Sweep (Cycle)"

```
d2=E0.1 / d1=E0.0 / g1=E0.3 / g2=E0.4 A0.0= Plus à gauche A0.1 = Plus à droite
```

[-] Réseau 1 : Pour G

```
UN  "d2"          E0.1
UN  "d1"          E0.0
U(
O    "g2"          E0.4
O    "g1"          E0.3
)
O
U(
ON   "d2"          E0.1
ON   "d1"          E0.0
)
U    "g2"          E0.4
U    "g1"          E0.3
=    "A0.0"        A0.0
```

[-] Réseau 2 : Pour D

```
U(
O    "d1"          E0.0
O    "d2"          E0.1
)
UN   "g1"          E0.3
UN   "g2"          E0.4
O
U    "d1"          E0.0
U    "d2"          E0.1
U(
ON   "g1"          E0.3
ON   "g2"          E0.4
)
=    "A0.1"        A0.1
```

[-] Réseau 3 : Egalité

```
UN   "A0.0"        A0.0
UN   "A0.1"        A0.1
=    "A0.2"        A0.2
```

Le langage LIST est, au demeurant, un peu plus compliqué à comprendre.

Il est cependant beaucoup plus compact que les deux autres et donc certainement plus rapide à exécuter.

Le programme se compose d'une suite de lignes, chacune spécifiant un opérateur logique suivi d'un opérande.

Opérande : adresse (E0.0, E0.1, E0.2, ...) ou une mnémonique entre "..."

Les opérateurs logiques sont codés de la manière suivante :

- U pour ET (und en allemand)
- O pour OU (oder en allemand)
- X pour OU EXCLUSIF
- N pour INVERSEUR (UN, ON)
- = la sortie (A0.0, A0.1, A0.2)

Attention : Il ne faut pas oublier les parenthèses dans les programmes ci-dessus

Exercice 5 : Séquentiel (Bascules)

Dans cet exercice, on étudie une porte de garage en se posant les questions suivantes :
Quand faut-il allumer et éteindre la montée ?
Quand faut-il allumer et éteindre la descente ?

On a déjà vu cet exercice en cours, on utilise donc la même démarche (bascules RS).

On définit les entrées suivantes :
o = on appuie sur le bouton ouvrir
f = on appuie sur le bouton fermer
s = on appuie sur le bouton stop
h = la porte de garage est en haut
b = la porte de garage est en bas

On définit les sorties suivantes :

SM = on allume la montée, la porte de garage monte
RM = on éteint la montée, la montée de la porte de garage est arrêtée
SD = on allume la descente, la porte de garage descend
RD = on éteint la descente, la descente de la porte de garage est arrêtée

On obtient les équations suivantes :

$$SM = o \cdot \bar{h}$$
$$RM = h + s$$

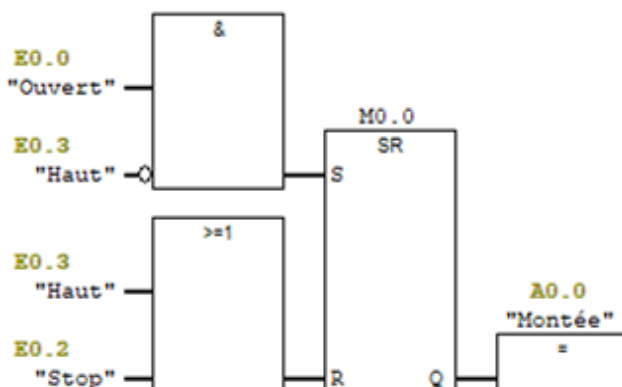
$$SD = f \cdot \bar{b}$$
$$RD = b + s$$

On programme ensuite ces équations en langage LOG (avec des mnémoniques on comprend beaucoup mieux ici) :

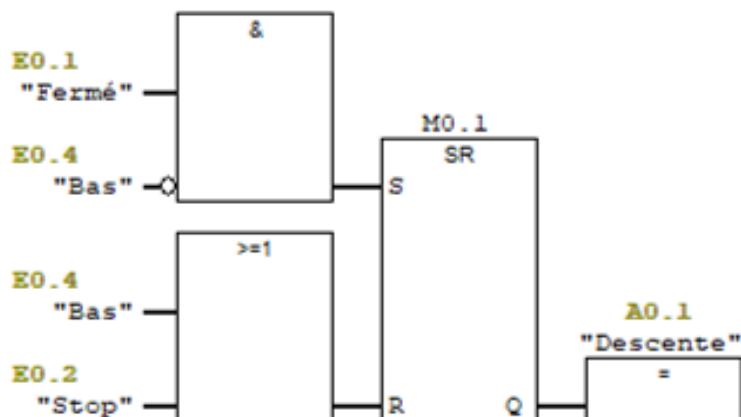
OBI : "Main Program Sweep (Cycle)"

Commentaire :

☐ Réseau 1 : Séquentiel Montée



Réseau 2 : Séquentiel Descente



On a quatre problèmes dans ces programmes :

- Quand on allume la montée (SM), **il ne faut pas appuyer sur fermer**.
Il faut appuyer sur ouvrir sans appuyer sur fermer en même temps et la porte ne doit pas être en haut.
- La montée de la porte s'arrête (RM) quand la porte est en haut ou quand on appuie sur le bouton stop ou encore quand **on appuie sur le bouton fermer** (la porte va alors descendre).
- Quand on allume la descente (SD), **il ne faut pas appuyer sur ouvrir**.
Il faut appuyer sur fermer sans appuyer sur ouvrir en même temps et la porte ne doit pas être en bas.
- La descente de la porte s'arrête (RD) quand la porte est en bas ou quand on appuie sur le bouton stop ou encore quand **on appuie sur le bouton ouvrir** (la porte va alors monter).

On obtient donc les équations suivantes :

$$SM = o \cdot \bar{h} \cdot \bar{f}$$

$$RM = h + s + f$$

$$SD = f \cdot \bar{b} \cdot \bar{o}$$

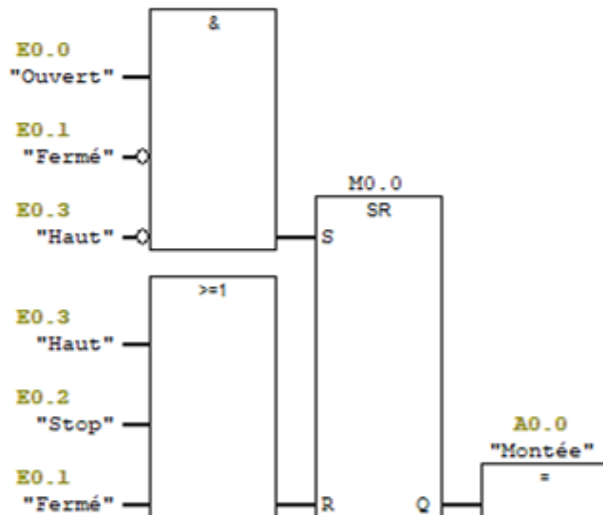
$$RD = b + s + o$$

On peut maintenant programmer ces corrections :

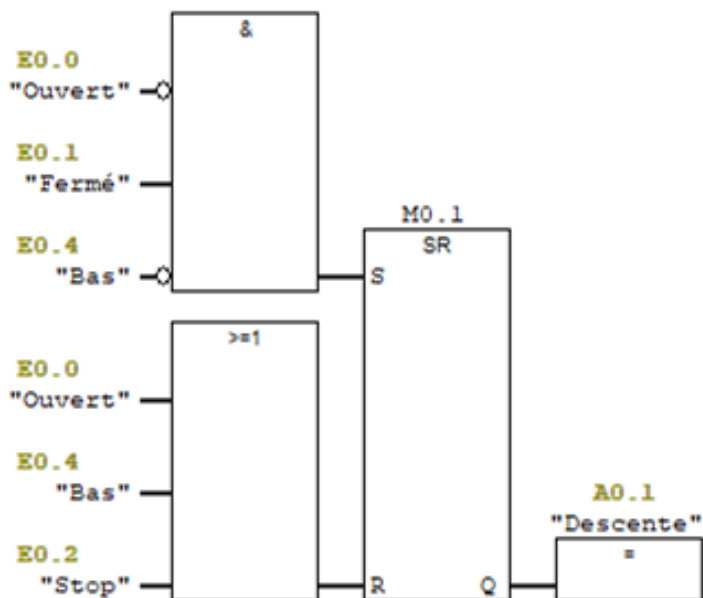
OBI : "Main Program Sweep (Cycle)"

Commentaire :

▣ Réseau 1 : Séquentiel Montée



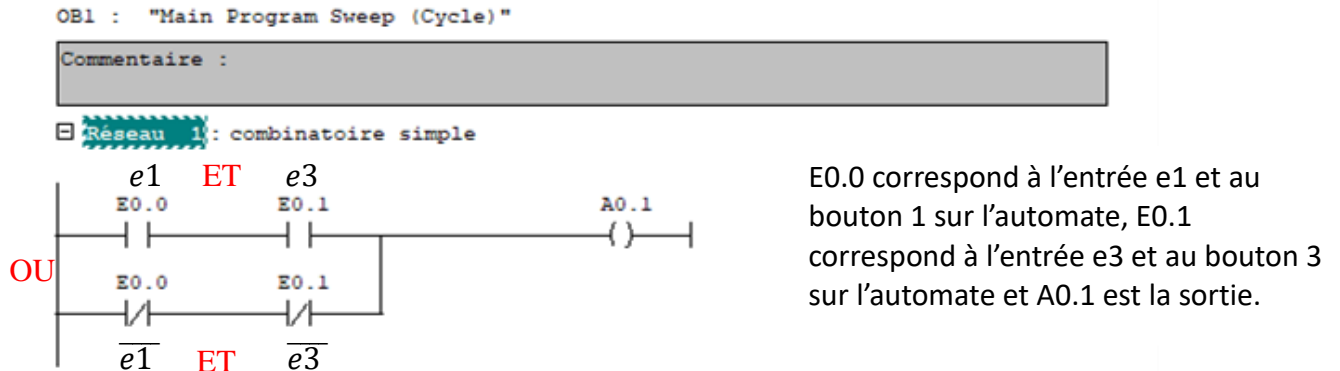
▣ Réseau 2 : Séquentiel Descente



Ces programmes ont semblé correct durant nos tests.

Annexes :

Exercice 1



Calcul pour l'exercice 2

$$\begin{aligned}
 PD &= \bar{g}_1 \bar{g}_2 d_2 + \bar{g}_1 d_1 d_2 + \bar{g}_2 d_1 d_2 + \bar{g}_1 \bar{g}_2 d_1 \\
 &= \bar{g}_1 \bar{g}_2 (d_1 + d_2) + d_1 d_2 (\bar{g}_1 + \bar{g}_2) \\
 PG &= g_1 g_2 \bar{d}_1 + g_1 g_2 \bar{d}_2 + \bar{d}_1 \bar{d}_2 g_2 + \bar{d}_1 \bar{d}_2 g_1 \\
 &= \bar{d}_1 \bar{d}_2 (g_1 + g_2) + g_1 g_2 (\bar{d}_1 + \bar{d}_2) \\
 E &= \overline{PD \cdot PG}
 \end{aligned}$$