



R119 – Fonctions de base de la logique Séquentielle

Licence Pro Rob&IA

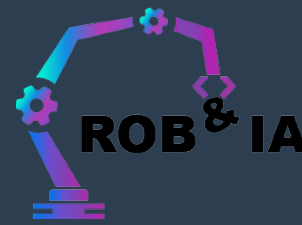
Laurent ROY

Table des matières



- Partie I : Temporisations
- **Partie II : Mémoires**
- Partie III : Compteurs

II-1 Différence entre la logique combinatoire et la logique séquentielle



1.1 Réalisation d'une régulation de niveau TOR en logique combinatoire

Logique d'activation du détecteur LW : $LW = 0$ si niveau L côté marquée.

XV et LW => variable booléennes (pas h)

Pour réaliser une régulation de niveau : la commande TOR de XV est décrite par sa **caractéristique** $XV = f(h)$ (h)

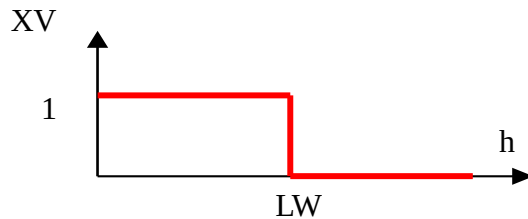
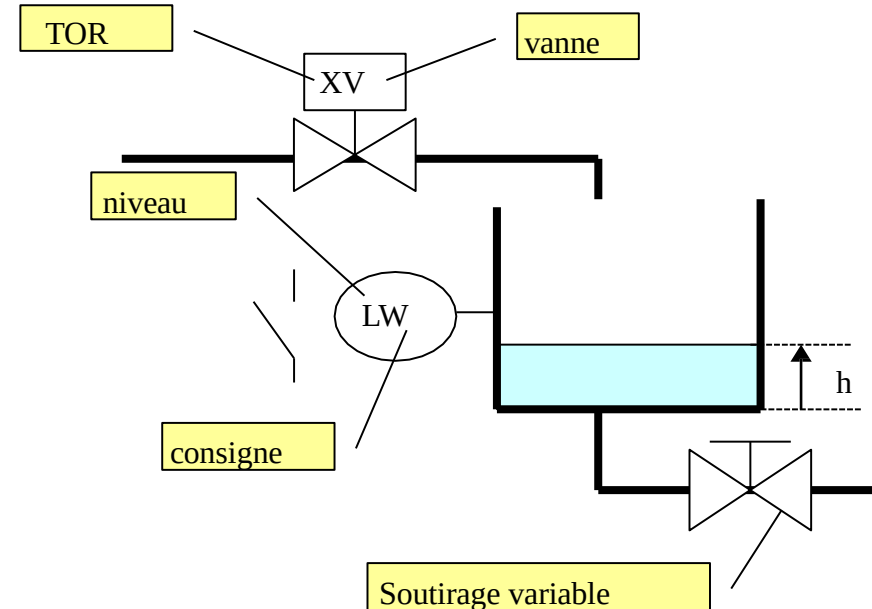
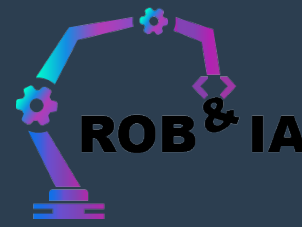


table de vérité

LW	XV
0	1
1	0



II-1 Différence entre la logique combinatoire et la logique séquentielle

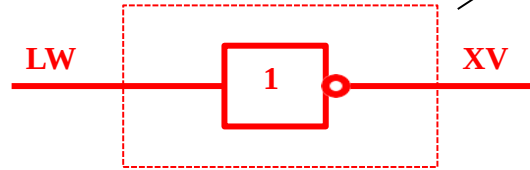


On voit que l'état de la sortie XV à un instant donné ne dépend que de l'état de l'entrée LW au même instant. => Logique combinatoire.

Equation logique :

$$XV = \overline{LW}$$

Logigramme :



Système combinatoire composé uniquement de fonction combinatoires (ET, OU, INV, ...) sans retour d'états

Conclusion sur cette commande en logique combinatoire :

A priori, grande précision sur la valeur réglée MAIS la fréquence de sollicitation de la vanne est très élevée.

Dans la pratique cette commande est inutilisable (longévité, fiabilité du dispositif !!!) mises à part certaines installations qui ont beaucoup d'inertie.

Exemple : thermostat à 1 seuil pour commander une chaudière collective. Il y a beaucoup d'inertie thermique ; en effet quand la commande passe à 1 :

- Temps de chauffe du corps de la chaudière
- Temps de chauffe du circuit d'eau.
- Temps de chauffe des radiateurs en fonte (la fonte est justement utilisée pour son inertie thermique).
- Temps de chauffe de l'air de la pièce.

II-1 Différence entre la logique combinatoire et la logique séquentielle

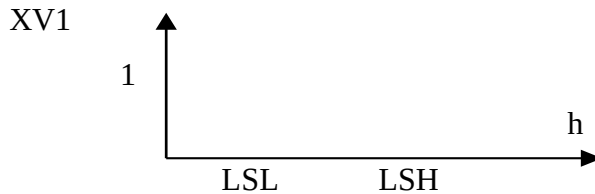


1.2 Amélioration de la régulation TOR : régulation TOR à hystérésis.

Cahier des charges : Début du remplissage au niveau LSL

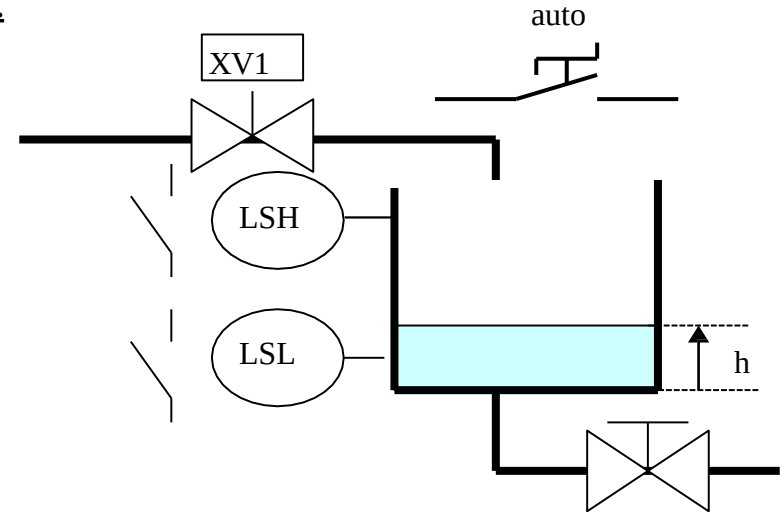
Fin du remplissage au niveau LSH

Caractéristique à hystérésis $XV1 = f(h)$



Equation logique :

Logigramme



II-1 Différence entre la logique combinatoire et la logique séquentielle

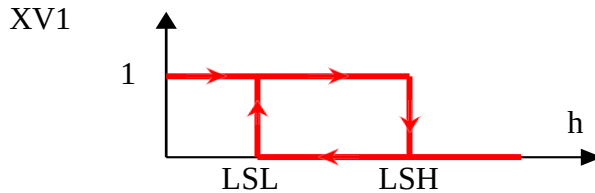


1.2 Amélioration de la régulation TOR : régulation TOR à hystérésis.

Cahier des charges : Début du remplissage au niveau LSL

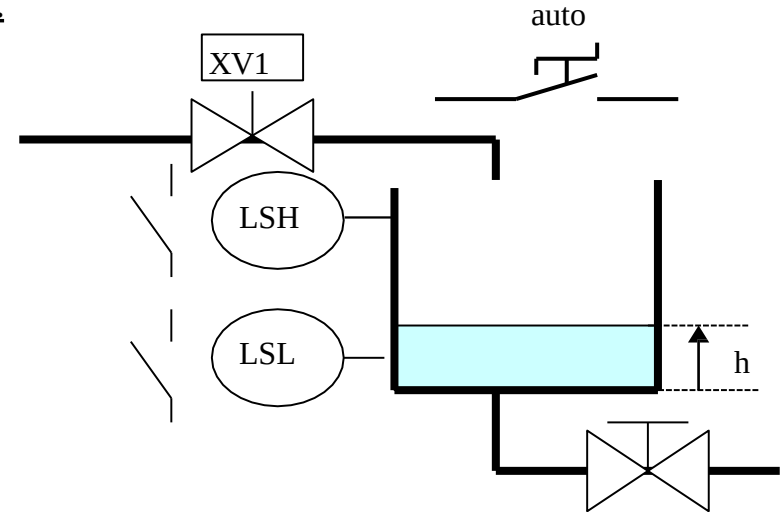
Fin du remplissage au niveau LSH

Caractéristique à hystérésis $XV1 = f(h)$



Equation logique :

Logigramme



II-1 Différence entre la logique combinatoire et la logique séquentielle

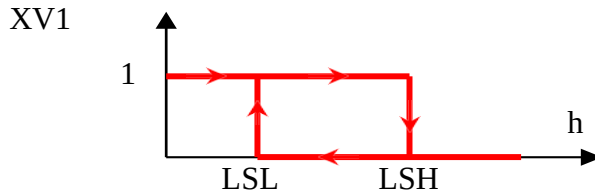


1.2 Amélioration de la régulation TOR : régulation TOR à hystérésis.

Cahier des charges : Début du remplissage au niveau LSL

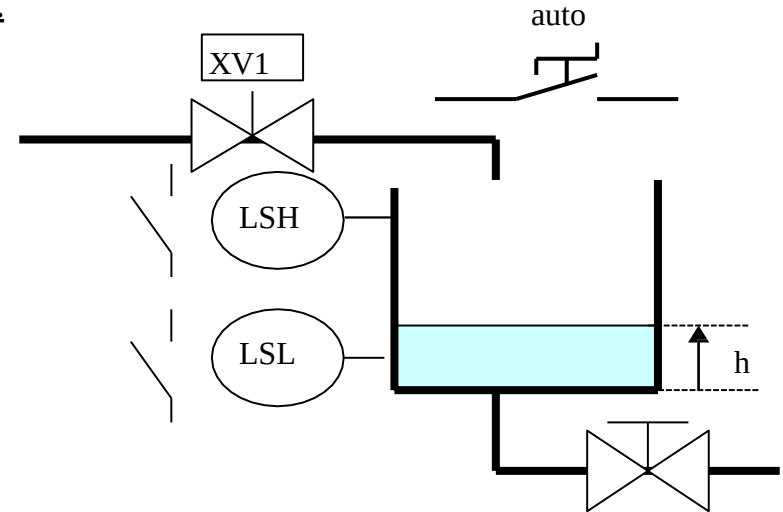
Fin du remplissage au niveau LSH

Caractéristique à hystérésis $XV1 = f(h)$

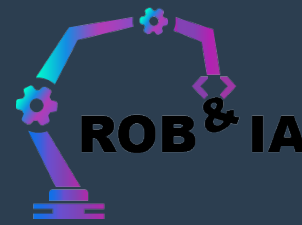


Equation logique : $XV1 = \overline{LSH} \cdot (\overline{LSL} + XV1)$

Logigramme



II-1 Différence entre la logique combinatoire et la logique séquentielle

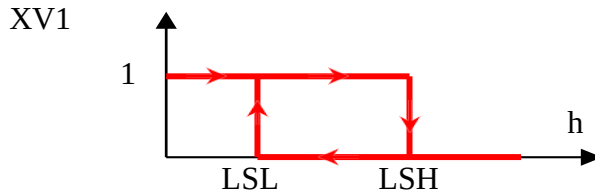


1.2 Amélioration de la régulation TOR : régulation TOR à hystérésis.

Cahier des charges : Début du remplissage au niveau LSL

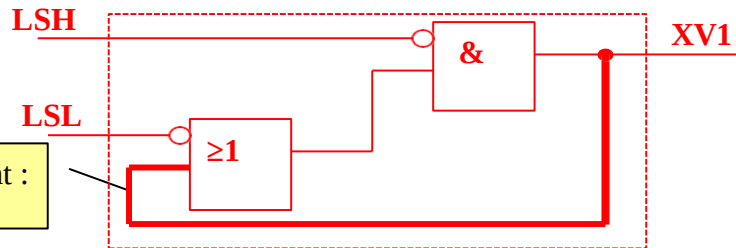
Fin du remplissage au niveau LSH

Caractéristique à hystérésis $XV1 = f(h)$

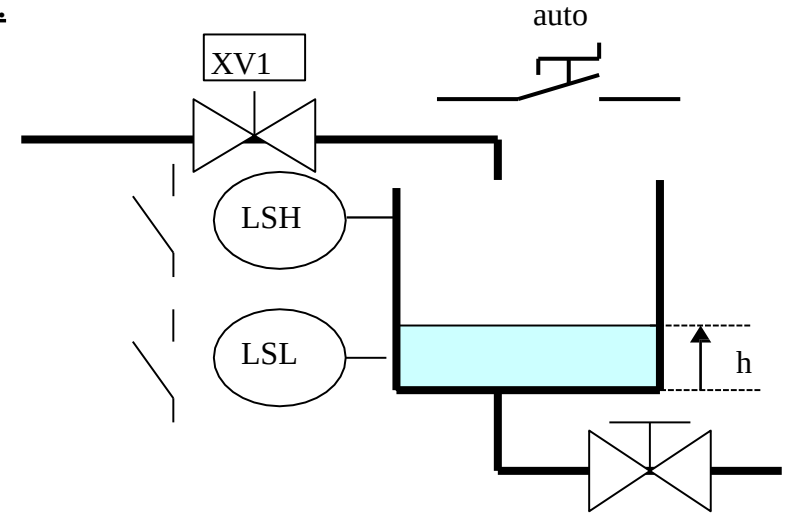


Equation logique : $XV1 = \overline{LSH} \cdot (\overline{LSL} + XV1)$

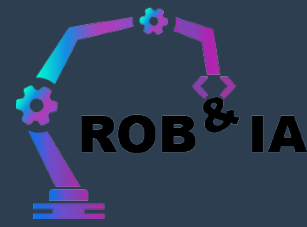
Logigramme



Variable interne d'état :
retour d'état

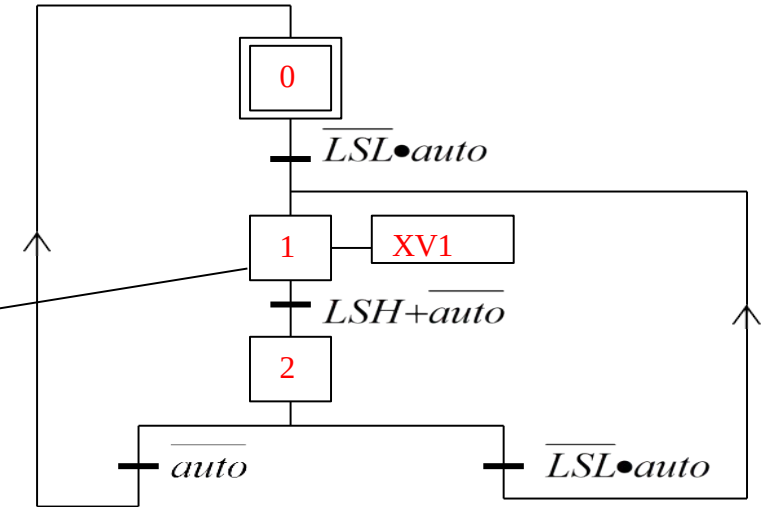


II-1 Différence entre la logique combinatoire et la logique séquentielle

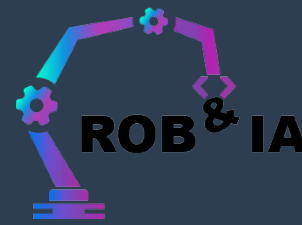


Le langage de programmation qui est particulièrement bien adapté à la logique séquentielle est le GRAFCET (GRAPhe Fonctionnel de Commande Etape Transition) Exemple : prise en compte du commutateur « auto »

L'état de XV1 dépend de $auto \wedge LSL$ mais aussi du n° d'étape active c'est à dire du passé du système



II-2 Mémoire (ou bascule) RS



2.1 Présentation

La mémoire RS est le composant élémentaire de la logique séquentielle. Il existe d'autres types de bascules (D, JK) mais qui ne sont pas utilisées dans les API.

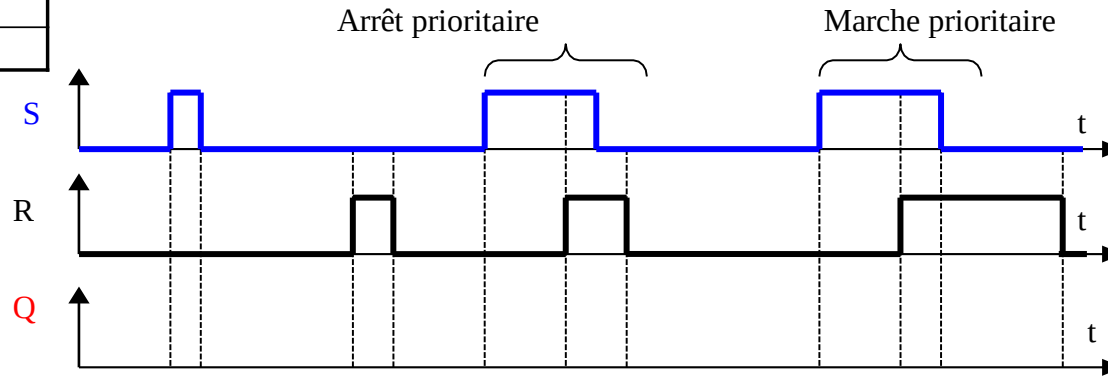
Mémoire RS : l'état de la sortie est conservée (mémorisée) en l'absence de sollicitation sur les deux entrées Set (mise à 1) et Reset (mise à 0)

S	R	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	*

Si la mémoire est à marche prioritaire alors $*$ = 1
Si la mémoire est à arrêt prioritaire alors $*$ = 0



Chronogramme



II-2 Mémoire (ou bascule) RS



2.1 Présentation

La mémoire RS est le composant élémentaire de la logique séquentielle. Il existe d'autres types de bascules (D, JK) mais qui ne sont pas utilisées dans les API.

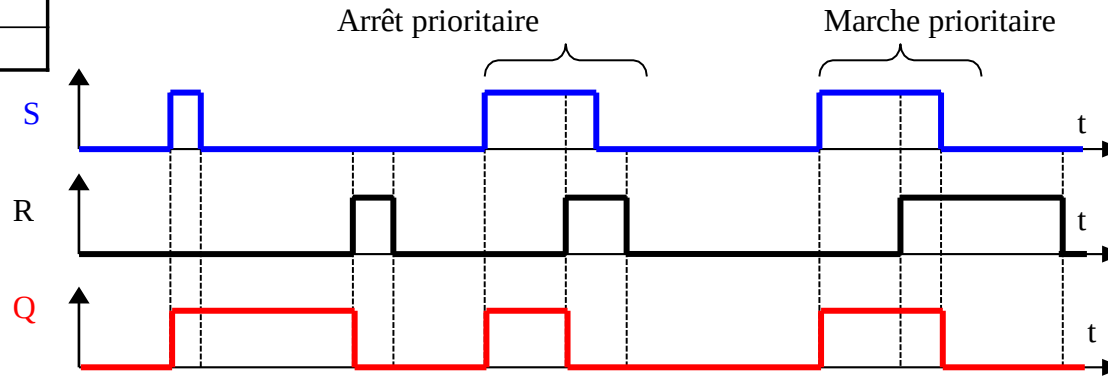
Mémoire RS : l'état de la sortie est conservée (mémorisée) en l'absence de sollicitation sur les deux entrées Set (mise à 1) et Reset (mise à 0)

S	R	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	*

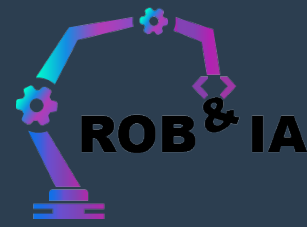
Si la mémoire est à marche prioritaire alors $*$ = 1
Si la mémoire est à arrêt prioritaire alors $*$ = 0



Chronogramme



II-2 Mémoire (ou bascule) RS



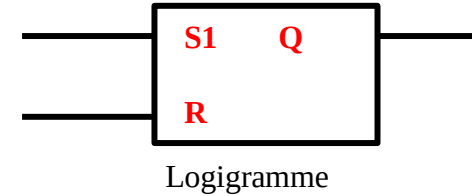
2.2 Mémoire à marche prioritaire (ou bascule à enclenchement prioritaire)

Cherchons Q en fonction de S et R

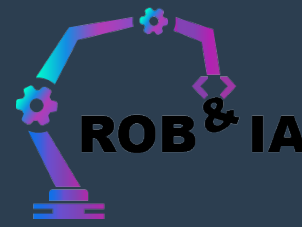
A partir de la table de vérité, on fait le tableau de Karnaugh

		S1			
		R			
Q _n	Q _{n-1}	00	01	11	10
	0	0	0	1	1
	1	1	0	1	1

L'équation simplifiée de cette mémoire RS₁ est



II-2 Mémoire (ou bascule) RS



2.2 Mémoire à marche prioritaire (ou bascule à enclenchement prioritaire)

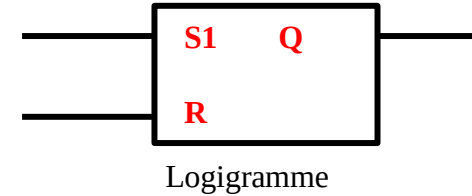
Cherchons Q en fonction de S et R

A partir de la table de vérité, on fait le tableau de Karnaugh

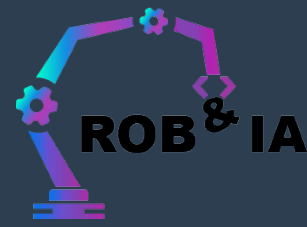
		S1			
		R			
Q _n	Q _{n-1}	00	01	11	10
	0	0	0	1	1
	1	1	0	1	1

Diagram illustrating the Karnaugh map for the RS flip-flop. The map shows the next state Q_n as a function of the current state Q_{n-1} and inputs S1 and R. Red boxes highlight the groups of 1s, with labels S1 and Q_{n-1} · R̄.

L'équation simplifiée de cette mémoire RS₁ est $Q_n = S + Q_{n-1} \cdot \bar{R}$



II-2 Mémoire (ou bascule) RS



2.2 Mémoire à marche prioritaire (ou bascule à enclenchement prioritaire)

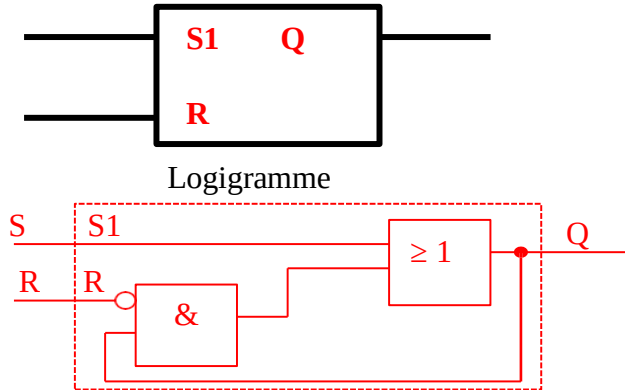
Cherchons Q en fonction de S et R

A partir de la table de vérité, on fait le tableau de Karnaugh

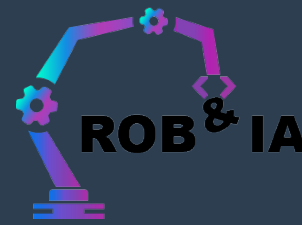
Q _n	R				S1
	00	01	11	10	
0	0	0	1	1	S1
1	1	0	1	1	
Q _{n-1}					

Diagram showing Karnaugh map groupings: a group of four cells (00, 01, 11, 10) for Q_n=0 is labeled S1; a group of four cells (00, 01, 11, 10) for Q_n=1 is labeled Q_{n-1} • R̄.

L'équation simplifiée de cette mémoire RS₁ est $Q_n = S + Q_{n-1} \cdot \bar{R}$



II-2 Mémoire (ou bascule) RS

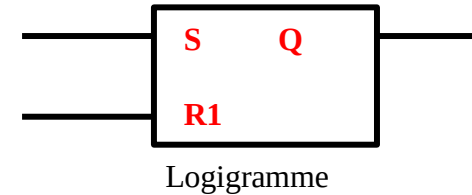


2.3 Mémoire à arrêt prioritaire (ou bascule à déclenchement prioritaire)

Cherchons Q en fonction de S et R

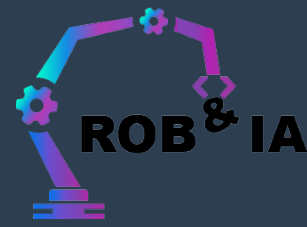
A partir de la table de vérité, on fait le tableau de Karnaugh

		<div><div></div><div></div><div></div><div></div></div> <div>S</div> <div>R1</div>			
Q _n		00	01	11	10
0		0	0	0	1
1		1	0	0	1
Q _{n-1}					



L'équation simplifiée de cette mémoire R₁S est

II-2 Mémoire (ou bascule) RS

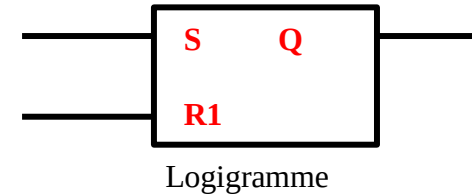


2.3 Mémoire à arrêt prioritaire (ou bascule à déclenchement prioritaire)

Cherchons Q en fonction de S et R

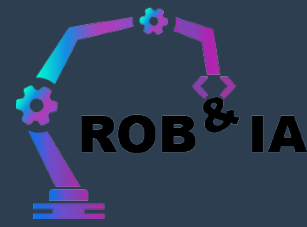
A partir de la table de vérité, on fait le tableau de Karnaugh

		S				
		R1				
Q _n		00	01	11	10	
	0	0	0	0	1	$S \cdot \overline{R1}$
	1	1	0	0	1	$Q_{n-1} \cdot \overline{R1}$
Q _{n-1}						



L'équation simplifiée de cette mémoire R₁S est $Q_n = \overline{R1} \cdot (S + Q_{n-1})$

II-2 Mémoire (ou bascule) RS



2.3 Mémoire à arrêt prioritaire (ou bascule à déclenchement prioritaire)

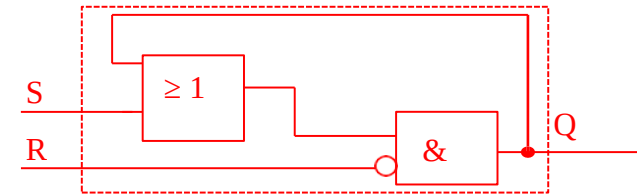
Cherchons Q en fonction de S et R

A partir de la table de vérité, on fait le tableau de Karnaugh

		S				
		R1				
Q _n		00	01	11	10	
0		0	0	0	1	$S \cdot \overline{R}1$
1		1	0	0	1	$Q_{n-1} \cdot \overline{R}1$
Q _{n-1}						

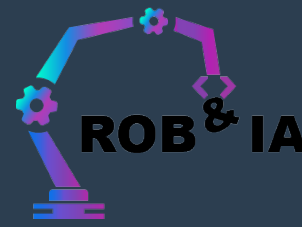


Logigramme

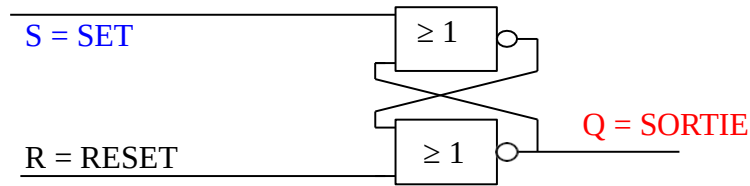


L'équation simplifiée de cette mémoire R₁S est $Q_n = \overline{R}1 \cdot (S + Q_{n-1})$

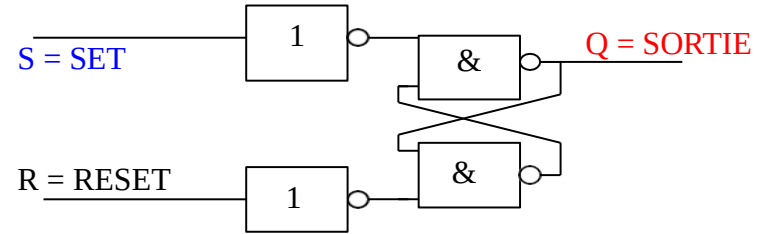
II-3 Réalisation pratiques de mémoire RS



3.1 En logique câblée (avec des portes NOR et NAND)



Arrêt prioritaire



Enclenchement prioritaire

3.2 En logique programmée (langage logigramme FBD)

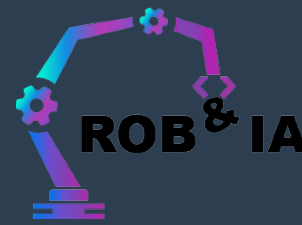


*Arrêt prioritaire
Bascule RS*



*Enclenchement prioritaire
bascule SR*

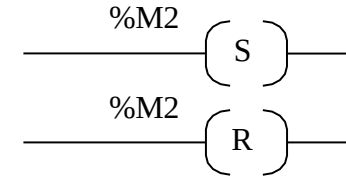
II-3 Réalisation pratiques de mémoire RS



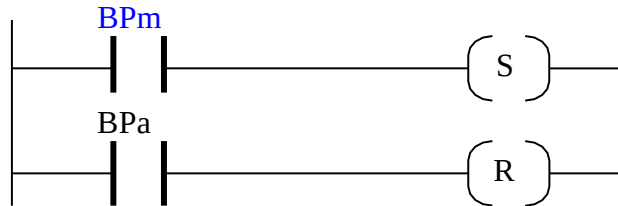
3.3 En logique programmée (langage à contacts LADDER)

En LADDER, il existe deux types de bobines :

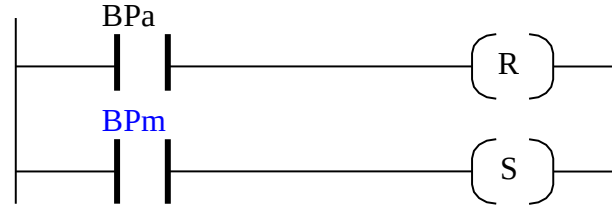
- Bobine SET, mise à 1 de %M2
- Bobine RESET, mise à 0 de %M2



!! C'est la dernière bobine qui est programmée qui est prioritaire !!

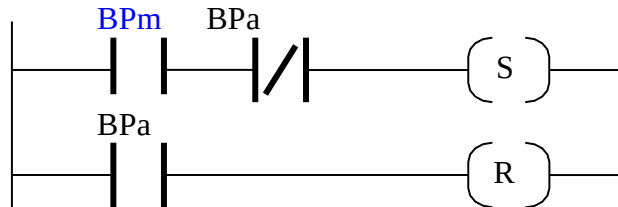


Arrêt prioritaire

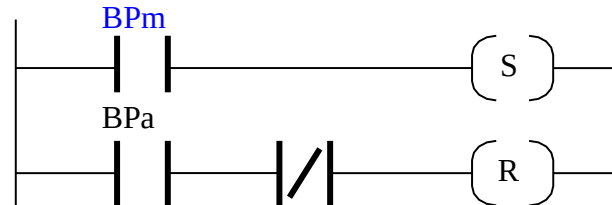


Enclenchement prioritaire

Pour éviter ce problème il faut rendre prioritaire PAR PROGRAMMATION la mémoire RS.

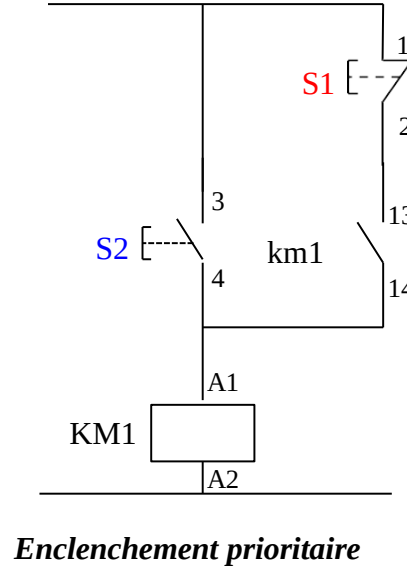
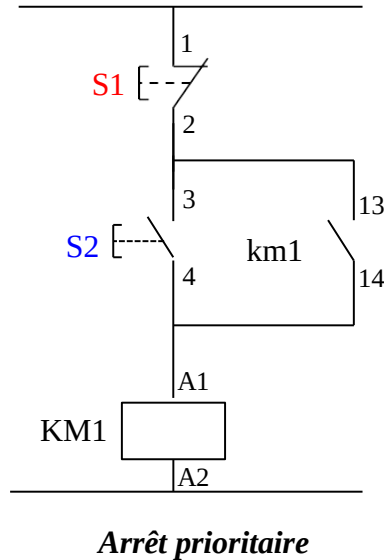
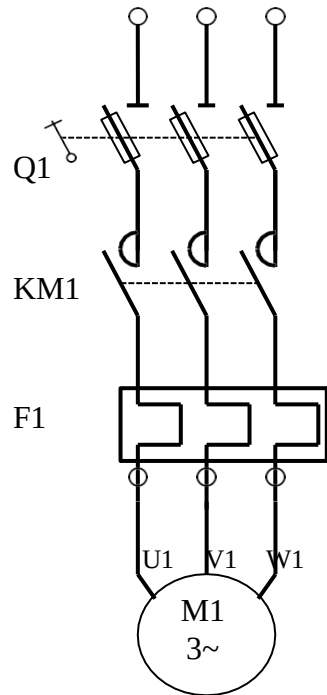
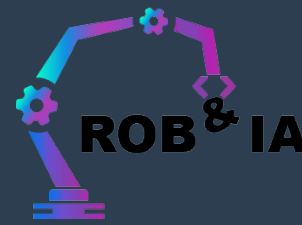


Arrêt prioritaire



Enclenchement prioritaire

II-3 Réalisation pratiques de mémoire RS



3.4 Avec un relais électromagnétique

Dans la commande d'un moteur par bouton poussoirs marche et arrêt, on réalise une fonction **MEMOIRE**, avec un relais électromagnétique. C'est ce que l'on appelle **l'AUTO-MAINTIENT**.

Ce montage permet lorsque l'on exerce une pression sur le bouton marche **S2**, on fait fonctionner le moteur et lorsque l'on lâche ce bouton, le moteur fonctionnera tant que l'on n'exercera pas une pression sur arrêt bouton poussoir **S1**.



Fin

Bibliographie :

Boujat G., Anaya P. : Automatique industrielle en 20 fiches, Dunod

Documentation constructeur Siemens :

<https://mall.industry.siemens.com>