



Intégrer un automate Siemens
S7-1200 ou S7-1500 dans un système
(STEP7 - TIA Portal)

- 06 - Les fonctions logiques

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

3

- LE SYSTEME COMBINATOIRE
- FONCTION « ET »
- FONCTION « OU inclusif »
- FONCTION « NON »
- FONCTION « NAND (NON ET)»
- FONCTION « NOR (NON OU)»
- FONCTION « OU exclusif»

2. EXERCICES

10

3. LES FONCTIONS LOGIQUES SEQUENTIELLES

11

- PRINCIPE DE FONCTIONNEMENT
- LE CHRONOGRAMME
- LA FONCTION « MÉMOIRE » AVEC PRIORITÉ A L'ACTIVATION
- LA FONCTION « MÉMOIRE » AVEC PRIORITÉ A LA DÉSACTIVATION

4. LES FRONTS MONTANTS ET DESCENDANTS

16

- LES FRONTS MONTANTS ET DESCENDANTS
- DETECTION DU FRONT MONTANT D'UN BIT
- DETECTION DU FRONT DESCENDANT D'UN BIT
- DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE
- DETECTION DES FRONTS MONTANTS ET DESCENDANTS AVEC DES BLOCS DE DONNÉES D'INSTANCE

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ LE SYSTEME COMBINATOIRE

- ✓ Un système est dit « combinatoire » lorsque la ou les sorties ne dépendent que de la combinaison des entrées.
- ✓ L'effet disparaît lorsque la cause disparaît.



institut des ressources
industrielles
- AFPI Lyon -

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ FONCTION « ET »

✓ Fonctionnement

➤ « Q0 » s'allume si LES DEUX boutons « S0 » et « S1 » sont actionnés simultanément.

✓ Table de vérité

S0	S1	Q0
0	0	0
0	1	0
1	0	0
1	1	1

$$\text{Equation: } Q0 = S0 \cdot S1$$

✓ Programmation en langage à contacts « CONT »:



✓ Programmation en logigramme « LOG »: &



✓ Programmation en liste d'instructions « LIST »: A: AND

1	A	"S0"	%I0.0
2	A	"S1"	%I0.1
3	=	"Q0"	%Q0.0

✓ Programmation en littéral structuré « SCL »: AND

```
1 (* Fonction "ET" *)
2 "Q0" := "S0" AND "S1";
```

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ FONCTION « OU inclusif »

✓ Fonctionnement

➤ « Q1 » s'allume si AU MOINS UN des deux boutons « S2 » et « S3 » est actionné.

✓ Table de vérité

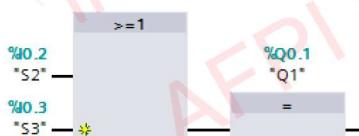
S2	S3	Q1
0	0	0
0	1	1
1	0	1
1	1	1

$$\text{Equation: } Q1 = S2 + S3$$

✓ Programmation en langage à contacts « CONT »:



✓ Programmation en logigramme « LOG »: >=1



✓ Programmation en liste d'instructions « LIST »: O: OR

1	O	"S2"		%I0.2
2	O	"S3"		%I0.3
3	=	"Q1"		%Q0.1

✓ Programmation en littéral structuré « SCL »: OR

```
4 (* Fonction "OU" *)
5 "Q1" := "S2" OR "S3";
```

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ FONCTION « NON »

✓ Fonctionnement

- « Q2 » s'allume si le bouton « S4 » n'est pas actionné.

✓ Table de vérité

S4	Q2
0	1
1	0

$$\text{Equation: } Q2 = \overline{S4}$$

✓ Programmation en langage à contacts « CONT »:

- 1^{ère} solution:

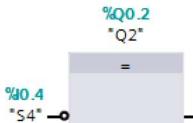


- 2^{ème} solution:

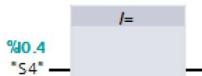


✓ Programmation en logigramme « LOG »:

- 1^{ère} solution:



- 2^{ème} solution: / =



✓ Programmation en liste d'instructions « LIST »:

- 1^{ère} solution: NOT

1	A	"S4"	
2	NOT		\$I0.4
3	=	"Q2"	\$Q0.2

- 2^{ème} solution: AN: AND NOT

1	A	"S4"	
2	=	"Q2"	\$Q0.2
3			

✓ Programmation en littéral structuré « SCL »: NOT

```
7 (* Fonction "NON" *)
8 "Q2" := NOT "S4";
```

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ FONCTION « NAND (NON ET) »

✓ Fonctionnement

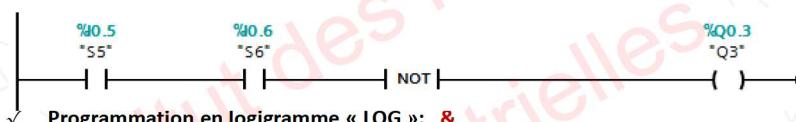
➤ « Q3 » s'allume si AUCUN BOUTON ou UN SEUL BOUTON est actionné.

✓ Table de vérité

S5	S6	Q3
0	0	1
0	1	1
1	0	1
1	1	0

$$\text{Equation: } Q3 = \overline{S5 \cdot S6}$$

✓ Programmation en langage à contacts « CONT »:



✓ Programmation en logigramme « LOG »: &



✓ Programmation en liste d'instructions « LIST »: A (AND) NOT

1	A	"S5"	\$I0.5
2	A	"S6"	\$I0.6
3	NOT		
4	=	"Q3"	\$Q0.3

✓ Programmation en littéral structuré « SCL »: NOTAND

```
10 (* Fonction "NAND (NON ET)" *)
11 "Q3" := NOT ("S5" AND "S6");
```

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ FONCTION « NOR (NON OU) »

✓ Fonctionnement

➤ « Q4 » s'allume si AUCUN BOUTON n'est actionné.

✓ Table de vérité

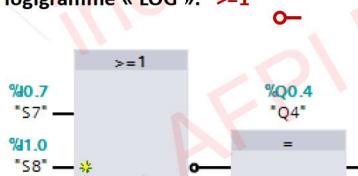
S7	S8	Q4
0	0	1
0	1	0
1	0	0
1	1	0

$$\text{Equation: } Q4 = \overline{S7 + S8}$$

✓ Programmation en langage à contacts « CONT »:



✓ Programmation en logigramme « LOG »:



✓ Programmation en liste d'instructions « LIST »:

1	O	"S7"	=\$I0.7
2	O	"S8"	=\$I1.0
3	NOT		
4	=	"Q4"	=\$Q0.4

✓ Programmation en littéral structuré « SCL »:

```
13 (* Fonction "NOR (NI)" *)
14 "Q4" := NOT ("S7" OR "S8");
```

Les fonctions logiques

1. LES FONCTIONS LOGIQUES COMBINATOIRES SUR BITS

□ FONCTION « OU exclusif »

✓ Fonctionnement

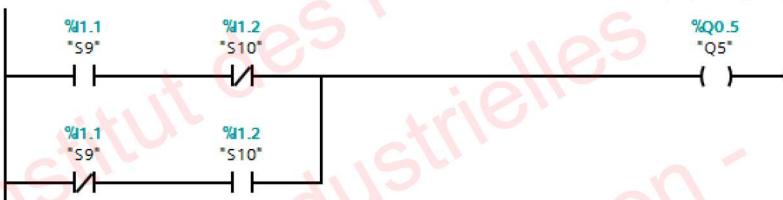
➤ « Q5 » s'allume si UN SEUL des deux boutons « S9 » et « S10 » est actionné.

✓ Table de vérité

S9	S10	Q5
0	0	0
0	1	1
1	0	1
1	1	0

Equation: $Q5 = S9 \oplus S10$
 $Q5 = \overline{S9} \cdot S10 + S9 \cdot \overline{S10}$

✓ Programmation en langage à contacts « CONT »:



✓ Programmation en logigramme « LOG »: X



✓ Programmation en liste d'instructions « LIST »: X

1	X	"S9"	=I1.1
2	X	"S10"	=I1.2
3	=	"Q5"	=Q0.5

✓ Programmation en littéral structuré « SCL »: NOTOR

```
16 (* Fonction "OU exclusif" *)
17 "Q5" := "S9" XOR "S10";
```

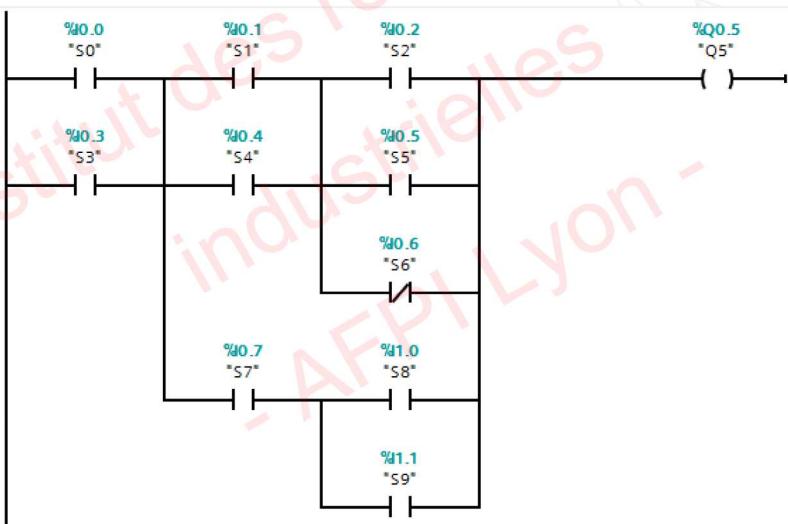
Les fonctions logiques

2. EXERCICES

✓ Programmer les équations suivantes dans les langages CONT, LOG, LIST et SCL:

- $Q0 = S0 \cdot S1 + S2$
- $Q1 = (S0 + S1 + S2) \cdot S3 \cdot /S4$
- $Q2 = S0 \cdot (/S1 + S2 + S3)$
- $Q3 = S0 \cdot S1 + S2 \cdot S3$
- $Q4 = (S0 + S2) \cdot (S1 + S3)$

✓ Programmer l'équation suivante dans les langages LOG, LIST et SCL:

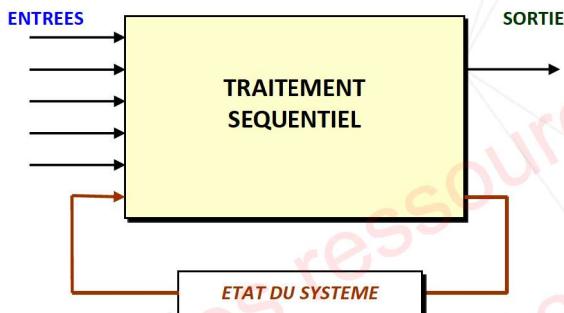


Les fonctions logiques

3. LES FONCTIONS LOGIQUES SEQUENTIELLES

□ PRINCIPE DE FONCTIONNEMENT

- ✓ La logique séquentielle, contrairement à la logique combinatoire fait intervenir le temps.
- ✓ L'état logique d'une variable de sortie à l'instant « t » dépend:
 - de l'état des entrées,
 - de l'état de l'installation à l'instant $t-1$



□ LE CHRONOGRAMME

- ✓ Le chronogramme est une représentation graphique de l'évolution temporelle d'une variable.
- ✓ On trouve sur l'axe des abscisses le temps, sur l'axe des ordonnées l'état logique 0 ou 1 de la variable étudiée



Les fonctions logiques

3. LES FONCTIONS LOGIQUES SEQUENTIELLES

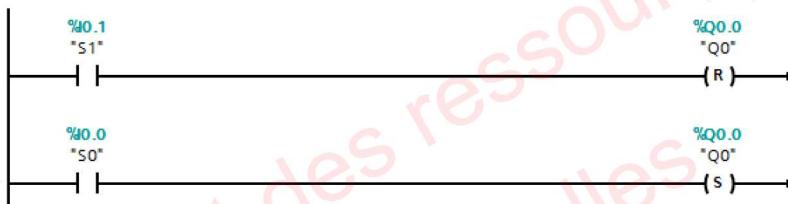
□ LA FONCTION « MÉMOIRE » AVEC PRIORITÉ A L'ACTIVATION

✓ Fonctionnement

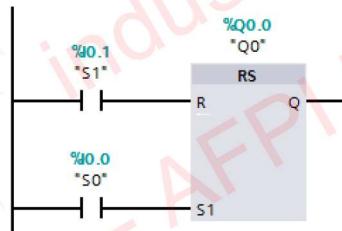
- Si on actionne « **S0** », « **Q0** » s'allume.
- Si on relâche le bouton « **S0** », « **Q0** » reste allumée.
- Si on actionne « **S1** », « **Q0** » s'éteint.
- Si on actionne simultanément les boutons « **S0** » et « **S1** », « **Q0** » est allumée.

✓ Programmation en langage à contacts « CONT »:

- **1^{ère} solution:** utilisation des bobines $\neg(s)$ et $\neg(r)$



- **2^{ème} solution:** utilisation d'une bascule RS



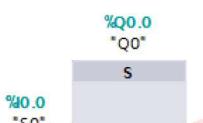
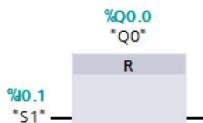
Les fonctions logiques

3. LES FONCTIONS LOGIQUES SEQUENTIELLES

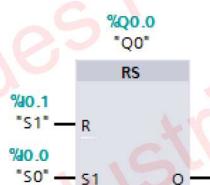
□ LA FONCTION « MÉMOIRE » AVEC PRIORITÉ A L'ACTIVATION

✓ Programmation en logigramme « LOG »:

➤ 1^{ère} solution: utilisation des commandes « R » et « S »



➤ 2^{ème} solution: utilisation d'une bascule RS



✓ Programmation en liste d'instructions « LIST »:

1	A	"S1"	\$I0.1
2	R	"Q0"	%Q0.0
3	A	"S0"	\$I0.0
4	S	"Q0"	%Q0.0

✓ Programmation en littéral structuré « SCL »:

```
1 (* "Mémoire avec priorité à l'activation" *)
2 IF "S1" THEN
3   "Q0" := 0;
4 END_IF;
5 IF "S0" THEN
6   "Q0":=1;
7 END_IF;
```

Les fonctions logiques

3. LES FONCTIONS LOGIQUES SEQUENTIELLES

□ LA FONCTION « MÉMOIRE » AVEC PRIORITÉ A LA DESACTIVATION

✓ Fonctionnement

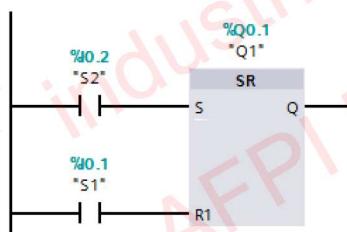
- Si on actionne « **S2** », « **Q1** » s'allume.
- Si on relâche le bouton « **S2** », « **Q1** » reste allumée.
- Si on actionne « **S1** », « **Q1** » s'éteint.
- Si on actionne simultanément les boutons « **S1** » et « **S2** », « **Q0** » est éteinte.

✓ Programmation en langage à contacts « CONT »:

- **1^{ère} solution:** utilisation des bobines $\neg(s)$ et $\neg(r)$



- **2^{ème} solution:** utilisation d'une bascule SR



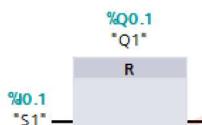
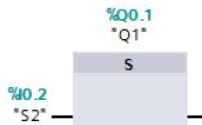
Les fonctions logiques

3. LES FONCTIONS LOGIQUES SEQUENTIELLES

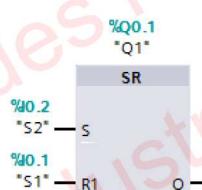
□ LA FONCTION « MÉMOIRE » AVEC PRIORITÉ A L'ACTIVATION

✓ Programmation en logigramme « LOG »:

➤ 1^{ère} solution: utilisation des commandes « R » et « S »



➤ 2^{ème} solution: utilisation d'une bascule SR



✓ Programmation en liste d'instructions « LIST »:

1	A	"S2"	\$I0.2
2	S	"Q1"	\$Q0.1
3	A	"S1"	\$I0.1
4	R	"Q1"	\$Q0.1

✓ Programmation en littéral structuré « SCL »:

```
9 (* "Mémoire avec priorité à la désactivation" *)
10 IF "S2" THEN
11   "Q1" := 1;
12 END_IF;
13 IF "S1" THEN
14   "Q1" := 0;
15 END_IF;
```

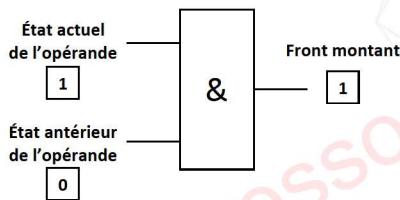
Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

□ DETECTION DU FRONT MONTANT D'UN BIT

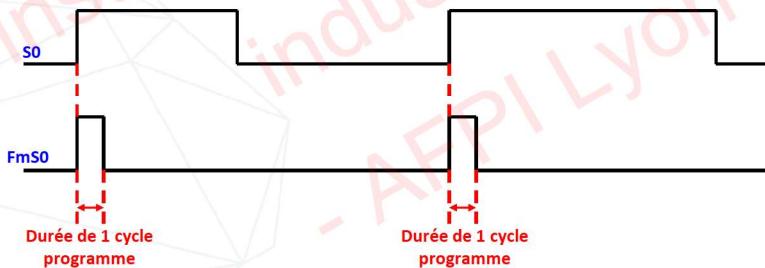
✓ Définition

- Un **front montant** est la détection du passage de l'état logique 0 à l'état logique 1 d'un opérande (bit) spécifié.
- L'instruction compare l'état logique actuel de l'**<opérande1>** à celui de l'interrogation précédente, mémorisé dans un memento de front (**<opérande2>**).



✓ Exemple:

- Lorsque « **S0** » passe à l'état 1, la variable « **FmS0** » passe à l'état 1 pendant la durée d'un cycle programme de l'automate.



Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

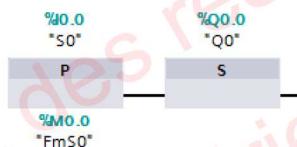
□ DETECTION DU FRONT MONTANT D'UN BIT

- ✓ Programmation en langage à contacts « CONT »:



❖ « Q0 » est activée par le front montant de « S0 ».

- ✓ Programmation en logigramme « LOG »:



- ✓ Programmation en liste d'instructions « LIST »:

1	A	"S0"	\$I0.0
2	FP	"FmS0"	\$M0.0
3	S	"Q0"	\$Q0.0

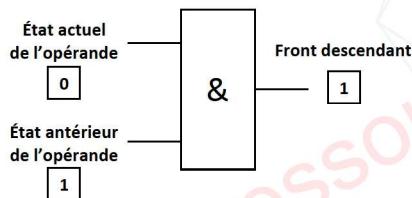
Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

□ DETECTION DU FRONT DESCENDANT D'UN BIT

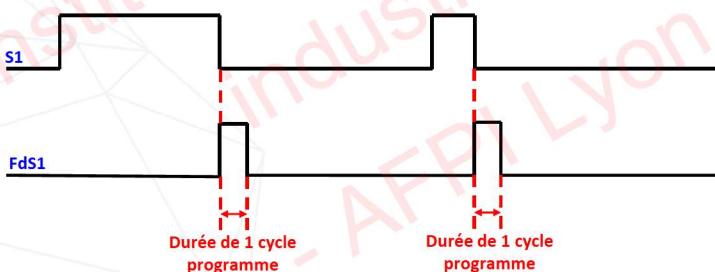
✓ Définition

- Un **front descendant** est la détection du passage de l'état logique 1 à l'état logique 0 d'un opérande (bit) spécifié.
- L'instruction compare l'état logique actuel de l'**<opérande1>** à celui de l'**interrogation précédente**, mémorisé dans un memento de front (**<opérande2>**).



✓ Exemple:

- Lorsque « **S1** » passe à l'état 0, la variable « **FdS1** » passe à l'état 1 pendant la durée d'un cycle programme de l'automate.

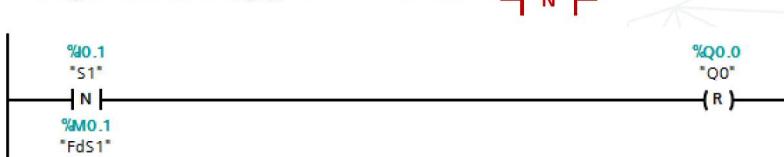


Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

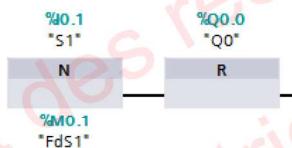
□ DETECTION DU FRONT DESCENDANT D'UN BIT

- ✓ Programmation en langage à contacts « CONT »:



❖ « **Q0** » est désactivée par le front descendant de « **S1** ».

- ✓ Programmation en logigramme « LOG »:



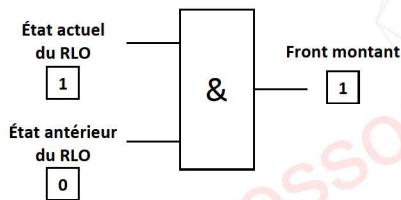
- ✓ Programmation en liste d'instructions « LIST »:

1	A	"S1"	\$I0.1
2	FN	"FdS1"	%M0.1
3	R	"Q0"	%Q0.0

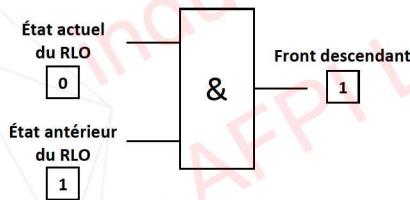
Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

- DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE
 - ✓ DETECTION DU FRONT MONTANT D'UNE EXPRESSION LOGIQUE
 - Cette instruction permet de mettre à l'état logique 1 un opérande si le registre de résultat logique (RLO) passe de l'état 0 à l'état 1.
 - Cette instruction compare le RLO actuel au RLO de l'interrogation précédente qui est mémorisé dans un mémento de front (<opérande2>).



- ✓ DETECTION DU FRONT DESCENDANT D'UNE EXPRESSION LOGIQUE
 - Cette instruction permet de mettre à l'état logique 1 un opérande si le registre de résultat logique (RLO) passe de l'état 1 à l'état 0.
 - Cette instruction compare le RLO actuel au RLO de l'interrogation précédente qui est mémorisé dans un mémento de front (<opérande2>).



Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE

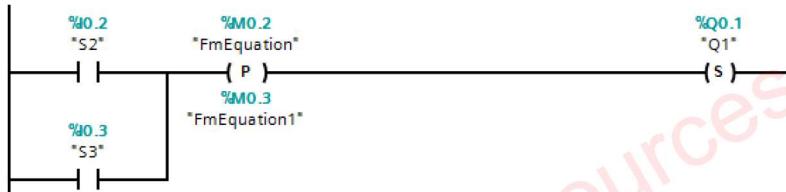
✓ PROGRAMMATION EN LANGAGE À CONTACTS « CONT »

➤ **1^{ère} solution:** front montant:

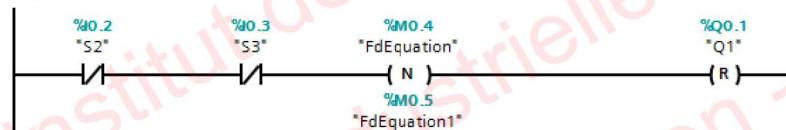
front descendant:

- (P)

- (N)



- ❖ « **Q1** » est activée lorsque le résultat de l'opération logique « **S2 + S3** » passe de l'état 0 à l'état 1 (« **%M0.2** » est activé pendant un cycle programme).
« **%M0.3** » contient le résultat de l'opération logique lors de l'exécution du cycle programme précédent.



- ❖ « **Q1** » est désactivée lorsque le résultat de l'opération logique « **S2 . S3** » passe de l'état 1 à l'état 0 (« **%M0.4** » est activé pendant un cycle programme).
« **%M0.5** » contient le résultat de l'opération logique lors de l'exécution du cycle programme précédent.

Les fonctions logiques

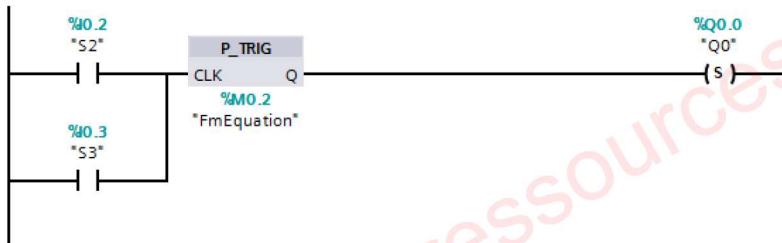
4. LES FRONTS MONTANTS ET DESCENDANTS

□ DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE

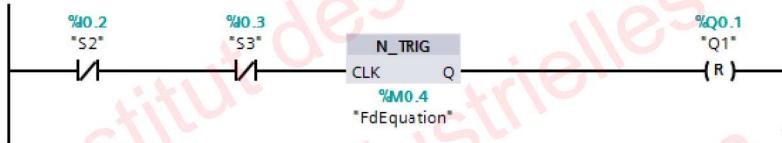
✓ PROGRAMMATION EN LANGAGE À CONTACTS « CONT »

➤ 2^{ème} solution: front montant: **P-TRIG** front descendant: **N-TRIG**

Réseau 5 : "Front montant d'une expression logique"



Réseau 6 : "Front descendant d'une expression logique"



- ❖ Les instructions « P-TRIG » et « N-TRIG » ne nécessitent pas l'utilisation d'un memento pour mémoriser le résultat de l'opération logique lors de l'exécution du cycle programme précédent.

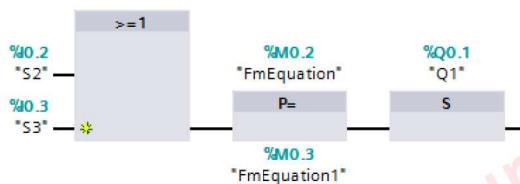
Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

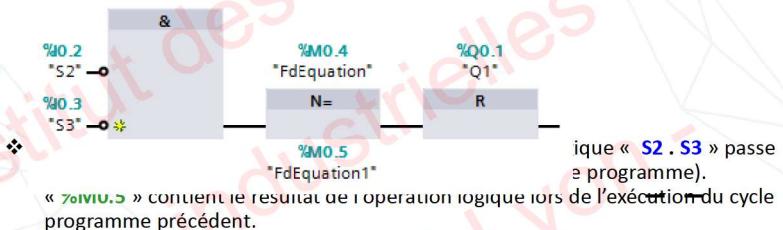
□ DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE

✓ PROGRAMMATION EN LOGIGRAMME « LOG »

➤ 1^{ère} solution: front montant: P= front descendant: N=



- ❖ « **Q1** » est activée lorsque le résultat de l'opération logique « **S2 + S3** » passe de l'état 0 à l'état 1 (« **%M0.2** » est activé pendant un cycle programme).
« **%M0.3** » contient le résultat de l'opération logique lors de l'exécution du cycle programme précédent.

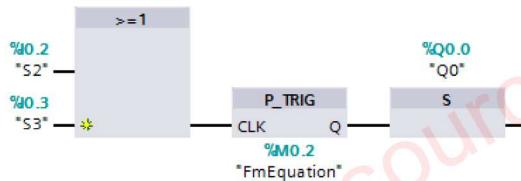


Les fonctions logiques

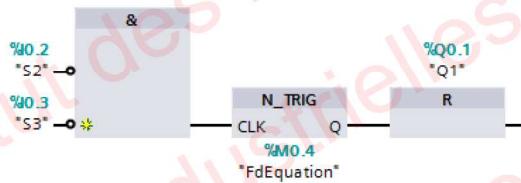
4. LES FRONTS MONTANTS ET DESCENDANTS

- DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE
 - ✓ PROGRAMMATION EN LANGAGE À CONTACTS « CONT »
 - 2^{ème} solution: front montant: **P-TRIG** front descendant: **N-TRIG**

Réseau 5 : "Front montant d'une expression logique"



Réseau 6 : "Front descendant d'une expression logique"



mémento pour mémoriser le résultat de l'opération logique lors de l'exécution du cycle programme précédent.

Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

- DETECTION DES FRONTS MONTANTS ET DESCENDANTS D'UNE EXPRESSION LOGIQUE
 - ✓ PROGRAMMATION EN LISTE D'INSTRUCTIONS « LIST »

➤ **1^{ère} solution:** front montant: **FP** front descendant: **FN**

1	O	"S2"	%IO.2
2	O	"S3"	%IO.3
3	FP	"FmEquation"	%MO.2
4	S	"Q1"	%OO.1

- ❖ « **Q1** » est activée lorsque le résultat de l'opération logique « **S2 + S3** » passe de l'état 0 à l'état 1 (« **%M0.2** » est activé pendant un cycle programme).

1	AN	"S2"	\$10.2
2	AN	"S3"	\$10.3
3	FN	"FdEquation"	\$M0.4
4	R	"O1"	\$00.1

- ❖ « **Q1** » est désactivée lorsque le résultat de l'opération logique « **S2 . S3** » passe de l'état 1 à l'état 0 (« **%M0.4** » est activé pendant un cycle programme).
« **%M0.5** » contient le résultat de l'opération logique lors de l'exécution du cycle programme précédent.

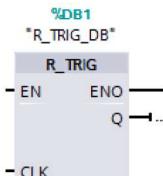
Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

□ DETECTION DES FRONTS MONTANTS ET DESCENDANTS AVEC DES BLOCS DE DONNEES D'INSTANCE

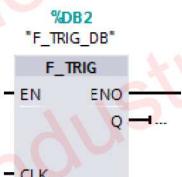
✓ LES INSTRUCTIONS R-TRIG / F-TRIG

➤ Front montant: R-TRIG



- ❖ L'instruction **R-TRIG** permet de détecter un changement d'état de "0" à "1" à l'entrée CLK. L'instruction compare l'état à l'entrée CLK à l'état de l'interrogation précédente (mémento de front), qui est enregistré dans l'instance indiquée.
- ❖ Lorsque l'instruction détecte un changement d'état de "0" à "1" à l'entrée CLK, un front montant est généré à la sortie Q.

➤ Front descendant: F-TRIG



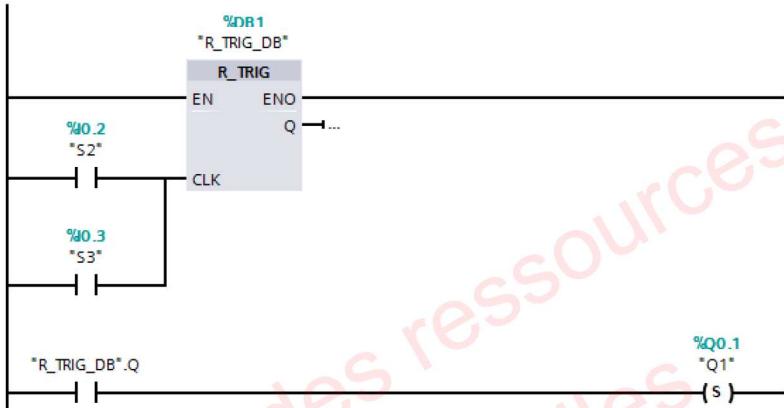
- ❖ L'instruction **R-TRIG** permet de détecter un changement d'état de "1" à "0" à l'entrée CLK. L'instruction compare l'état à l'entrée CLK à l'état de l'interrogation précédente (mémento de front), qui est enregistré dans l'instance indiquée.
- ❖ Lorsque l'instruction détecte un changement d'état de "1" à "0" à l'entrée CLK, un front descendant est généré à la sortie Q.

Les fonctions logiques

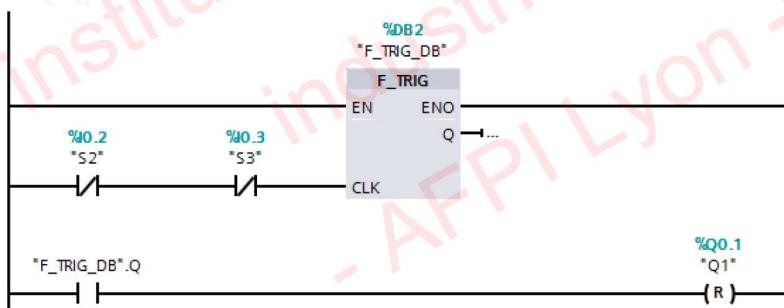
4. LES FRONTS MONTANTS ET DESCENDANTS

- DÉTECTION DES FRONTS MONTANTS ET DESCENDANTS AVEC DES BLOCS DE DONNÉES D'INSTANCE

- PROGRAMMATION EN LANGAGE À CONTACTS « CONT »



➤ « Q1 » est activée lorsque le résultat de l'opération logique « S2 + S3 » passe de l'état 0 à l'état 1.



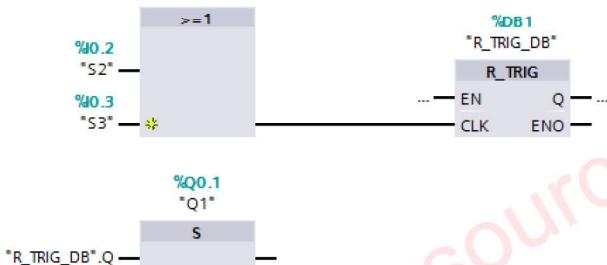
➤ « Q1 » est désactivée lorsque le résultat de l'opération logique « S2 . S3 » passe de l'état 1 à l'état 0.

Les fonctions logiques

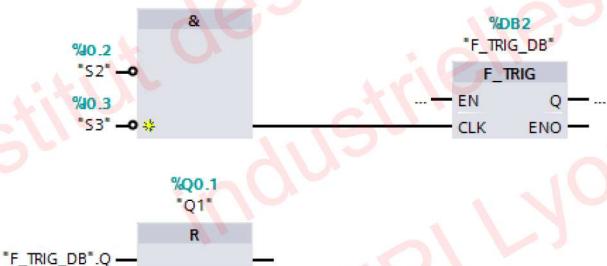
4. LES FRONTS MONTANTS ET DESCENDANTS

- DETECTION DES FRONTS MONTANTS ET DESCENDANTS AVEC DES BLOCS DE DONNEES D'INSTANCE

- ✓ PROGRAMMATION EN LOGIGRAMME « LOG »



- « Q1 » est activée lorsque le résultat de l'opération logique « S2 + S3 » passe de l'état 0 à l'état 1.



- « Q1 » est désactivée lorsque le résultat de l'opération logique « S2 . S3 » passe de l'état 1 à l'état 0.

Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

□ DETECTION DES FRONTS MONTANTS ET DESCENDANTS AVEC DES BLOCS DE DONNEES D'INSTANCE

✓ PROGRAMMATION EN LISTE D'INSTRUCTIONS « LIST »

- En langage « liste d'instructions », les instructions R-TRIG et F-TRIG ne peuvent détecter que le front montant ou descendant d'un opérande.

1	O	"S2"	\$I0.2
2	O	"S3"	\$I0.3
3	=	"R_TRIG_DB_2".Stat_Bit	
4	CALL	R_TRIG , "R_TRIG_DB_2"	\$DB4
5		CLK := "R_TRIG_DB_2".Stat_Bit	
6		Q :=	
7	A	"R_TRIG_DB".Q	
8	S	"Q1"	\$Q0.1

- « Q1 » est activée lorsque le résultat de l'opération logique « S2 + S3 » passe de l'état 0 à l'état 1.

1	AN	"S2"	\$I0.2
2	AN	"S3"	\$I0.3
3	=	"F_TRIG_DB_1".Stat_Bit	
4	CALL	F_TRIG , "F_TRIG_DB_1"	\$DB5
5		CLK := "F_TRIG_DB_1".Stat_Bit	
6		Q :=	
7	A	"F_TRIG_DB_1".Q	
8	R	"Q1"	\$Q0.1

- « Q1 » est désactivée lorsque le résultat de l'opération logique « S2 . S3 » passe de l'état 1 à l'état 0.

Les fonctions logiques

4. LES FRONTS MONTANTS ET DESCENDANTS

□ DETECTION DES FRONTS MONTANTS ET DESCENDANTS AVEC DES BLOCS DE DONNEES D'INSTANCE

✓ PROGRAMMATION EN LITTÉRAL STRUCTURÉ « SCL »

```
1 (*Front montant d'un bit ou d'une expression logique*)
2 R_TRIG_DB_1(CLK:="S2" OR "S3",
3 [ Q=>"FmEquation");
4 IF "FmEquation" THEN
5   "Q1" := 1;
6 END_IF;
```

- « **Q1** » est activée lorsque le résultat de l'opération logique « **S2 + S3** » passe de l'état 0 à l'état 1.

```
8 (*Front descendant d'un bit ou d'une expression logique*)
9 F_TRIG_DB_2(CLK:=NOT ("S2" OR "S3"),
10 [ Q=>"F_TRIG_DB_2".Q);
11 IF "F_TRIG_DB_2".Q THEN
12   "Q1" := 0;
13 END_IF;
```

- « **Q1** » est désactivée lorsque le résultat de l'opération logique « **S2 + S3** » passe de l'état 1 à l'état 0.