# Document de formation
# pour une solution complète d'automatisation
# Totally Integrated Automation (T I A)

## *ANNEXE III*

## Instructions de programmation de base

## CONT/LOG/LIST dans STEP 7

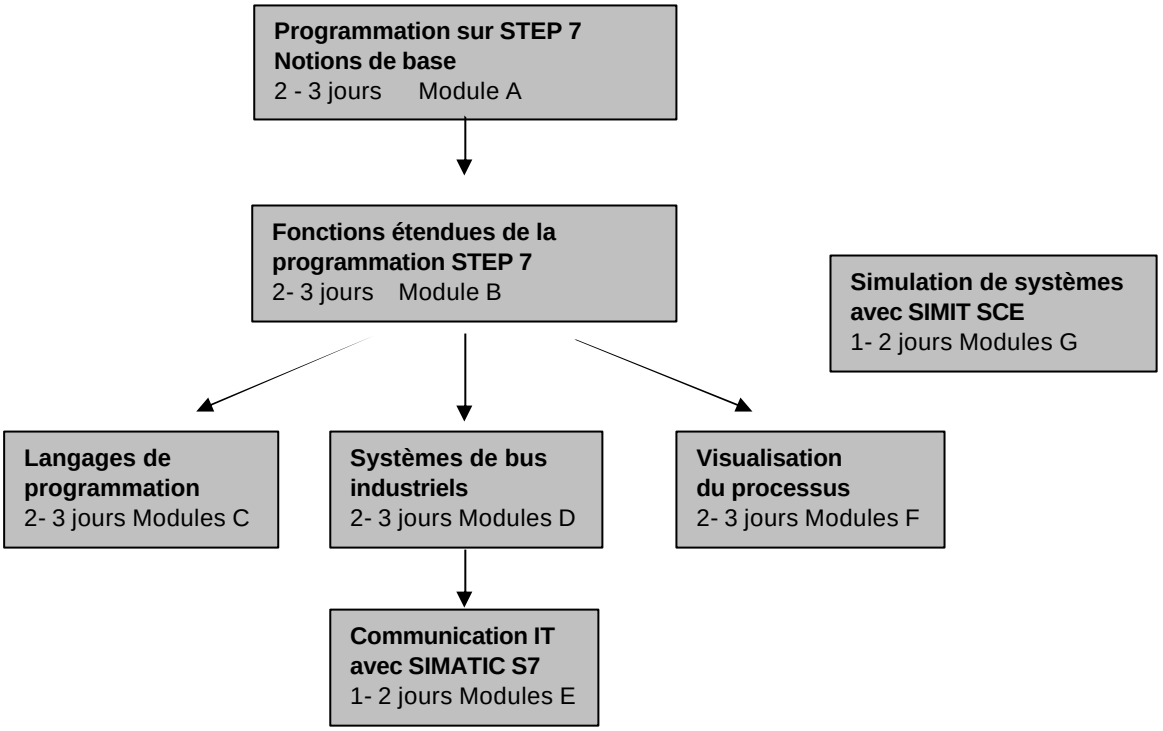
Nous remercions l'entreprise Michael Dziallas Engineering et les enseignants d'écoles professionnelles ainsi que tous ceux qui ont participé à l'élaboration de ce document.

PAGE :

PAGE :

## 1. AVANT-PROPOS

L'annexe III est nécessaire pour le traitement de tous les modules.




**Objectif pédagogique:**

Dans cette annexe, on vous présentera un ensemble de commandes de programmation importantes dont on se sert pour la résolution des exercices de programmation des modules.

**Prérequis :**

Afin de pouvoir comprendre les commandes et la manière de programmer, les connaissances suivantes doivent être acquises :

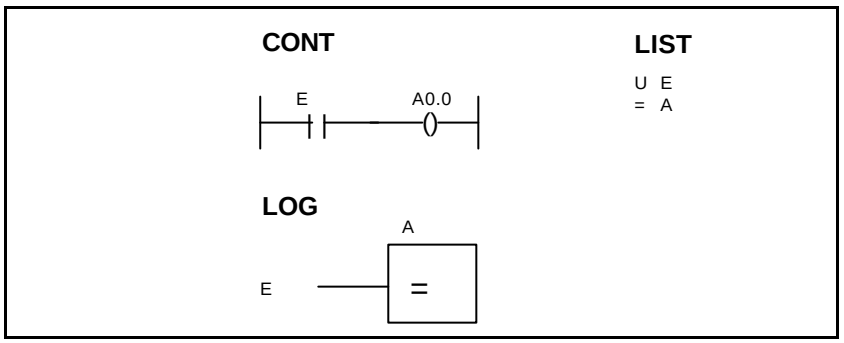
- Bases de la programmation d'automates (par ex. Annexe I – Bases de la programmation des API avec SIMATIC S7-300)

## 2. COMMANDES DE PROGRAMMATION DE BASE

Les commandes de programmation que l'on va vous présenter dans la suite sont suffisantes pour une programmation de base. Ce n'est en aucun cas une liste exhaustive des commandes.
Vous trouverez des informations supplémentaires sur d'autres commandes en CONT/LOG/LIST dans les manuels d'utilisation ou mieux, dans l'**aide en ligne** sous le point **Description du langage CONT, LOG et LIST**.

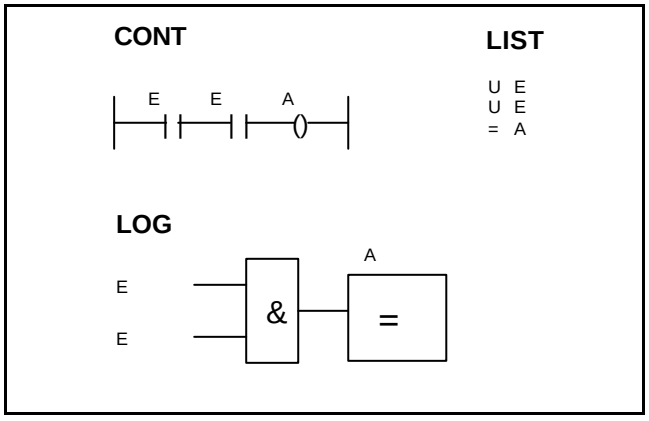
### 2.1 ATTRIBUTION

L'attribution (=) copie le résultat de la dernière opération logique (VKE) et l'attribue à l'opérande qui la suit.
Une chaîne logique peut être terminée par une attribution.



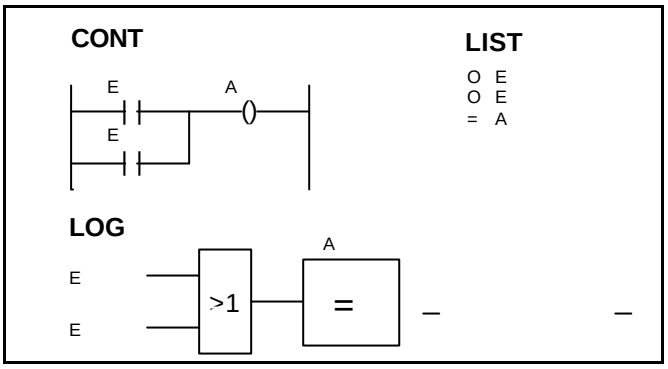

### 2.2 ET LOGIQUE

Le ET logique correspond à un montage de circuits électriques en série. En sortie A 0.0 s'obtient l'état de signal 1, si toutes les entrées sont simultanément à l'état 1. Si l'une des entrées bascule à l'état de signal 0, la sortie passe à l'état de signal 0.



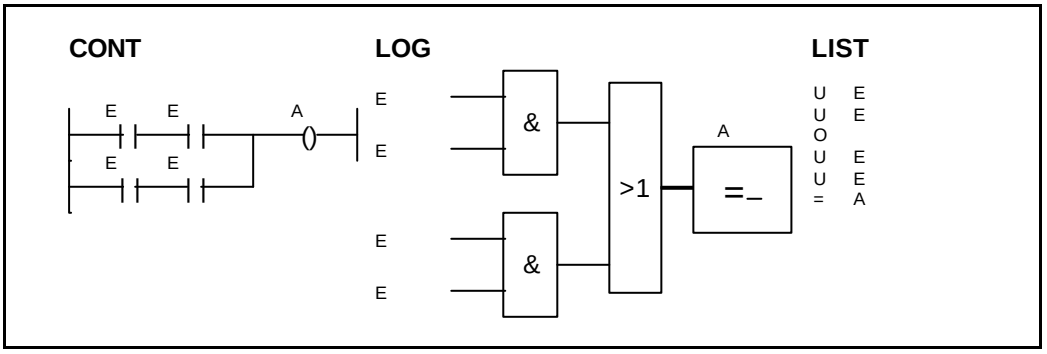
## 2.3 OU LOGIQUE

Le OU logique correspond à un montage de circuits électriques en parallèle. En sortie A 0.1 s'obtient l'état de signal 1, si au moins une entrée est à l'état 1. Si toutes les entrées basculent à l'état de signal 0, la sortie passe à l'état de signal 0.




## 2.4 ET AVANT OU LOGIQUE

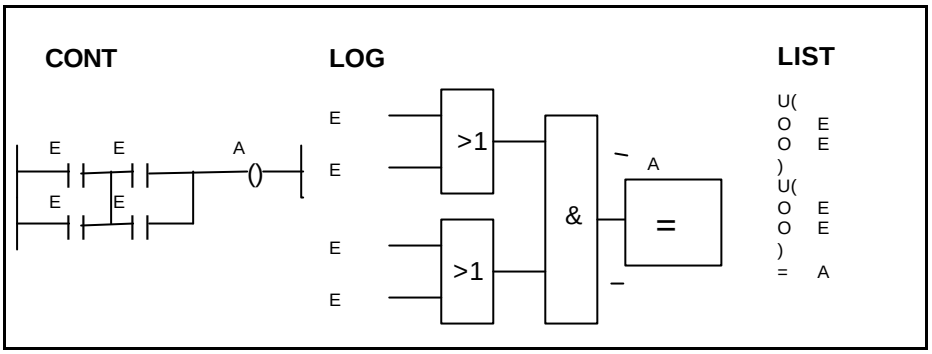
Le ET avant OU logique correspond à un montage de circuits électriques en parallèle de contacteurs en série.
Dans ce montage de branchements à la fois série et parallèle, la sortie 0.1 présente le signal d'état 1, si, dans au moins un branchement, tous les contacts branchés en série sont fermés (c.-à-d. qu'ils ont le signal d'état 1).
Le ET avant OU logique est programmé en mode LIST sans parenthèses, toutefois les branchements parallèles doivent être séparés l'un de l'autre par la saisie du caractère O (Fonction OU).
Dans cette opération, les fonctions ET sont traitées en premier, et ensuite leurs résultats sont traités en tant qu'entrées de la fonction OU. La première fonction ET (E 0.0, E 0.1) est séparée de la deuxième (E 0.2, E 0.3) par le caractère O (Fonction OU).




*Les opérations logiques ET ont priorité et seront donc traitées avant les opérations logiques OU.*

## 2.5 OU AVANT ET LOGIQUE

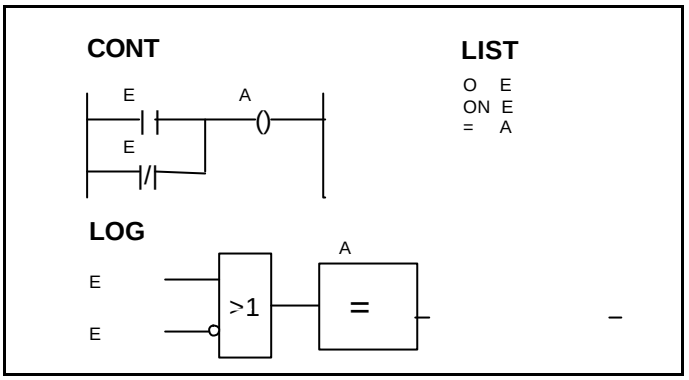
Le OU avant ET logique correspond à un montage de circuits électriques en série de contacts en parallèle.
Dans ce montage de branchements à la fois série et parallèle, la sortie 1.0 présente le signal d'état 1, seulement si, dans chaque branchement parallèle, au moins un des contacts est à l'état de signal 1.




*Afin que les opérations logiques OU aient priorité sur les opérations logiques ET, elles doivent être mises entre parenthèses.*

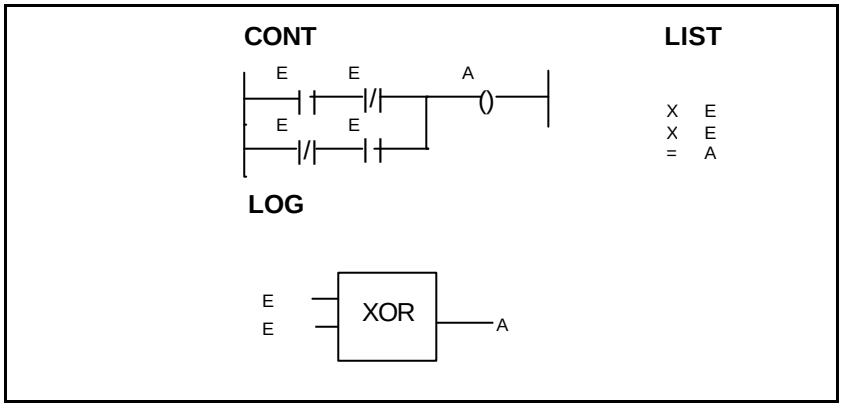
## 2.6 INTERROGATION DE L'ETAT DU SIGNAL 0

L'interrogation de l'état du signal 0 correspond à un contact à ouverture dans un circuit à contacteur et est réalisée par l'assemblage de : ET NON (UN), OU NON (ON) et NON OU EXCLUSIF (XN).
Exemple d'une opération logique OU NON :




## 2.7 OU EXCLUSIF LOGIQUE

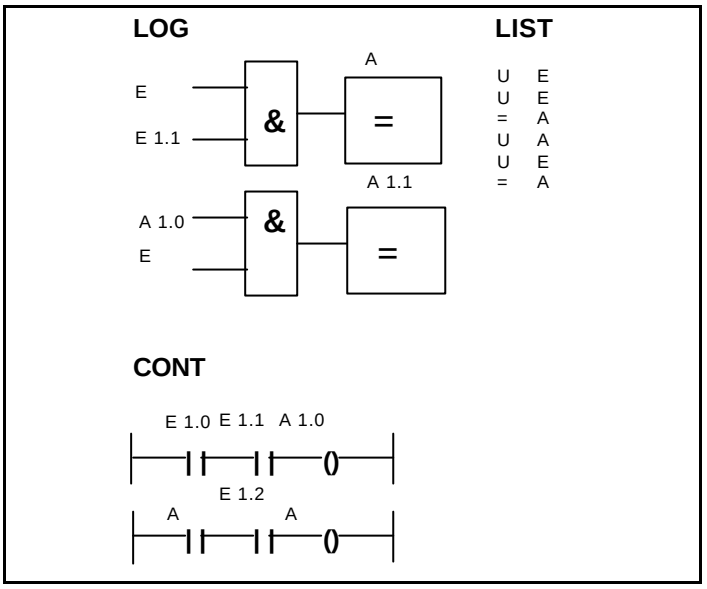
Le schéma électrique suivant présente un OU exclusif (X), dans lequel la sortie 1.0 n'est connectée (Etat de signal 1) que si les entrées livrent un état de signal à 1. Dans un circuit à contacteurs, cela peut être seulement réalisé à l'aide de contacts à ouverture et de contacts à fermeture.




**Indication :** *L'opération logique OU exclusif ne peut être employée qu'avec exactement deux entrées.*

## 2.8 INTERROGATION DES SORTIES

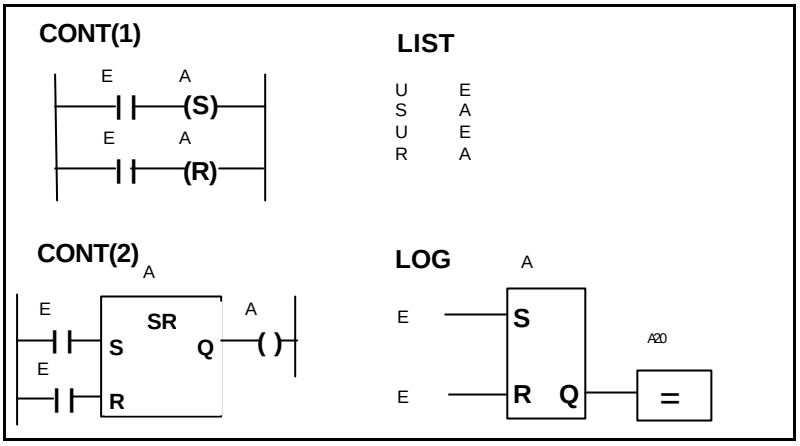
Certaines conditions doivent être respectées pour la connexion des sorties A 1.0 et A 1.1. Dans ces circonstances, on doit prévoir, pour chaque sortie, sa propre branche de circuit et son propre symbole logique. Puisque l'automate peut non seulement interroger l'état de signal des entrées mais aussi celui des sorties, des mémoires internes, etc., la boîte logique ET, pour la sortie A 1.1, interroge la sortie A 1.0.




## 2.9 FONCTIONS MEMOIRE RS

Conformément à DIN 40900 et DIN 19239, une fonction mémoire RS est représentée comme un rectangle avec l'entrée d'initialisation *S* et celle de réinitialisation *R*.
Un état bref de signal à 1 à l'entrée d'initialisation initialise la fonction mémoire. Un état bref de signal à 1 à l'entrée de réinitialisation réinitialise la fonction mémoire. L'état de signal 0 aux entrées R et S ne modifie pas l'état précédemment établi.
S'il se trouvait que les deux entrées R et S prennent toutes les deux un état de signal à 1, une priorité est établie à l'initialisation ou à la réinitialisation. Cette réinitialisation ou initialisation prioritaire doit être prise en compte dans votre programmation.

### 2.9.1 REINITIALISATION PRIORITAIRE

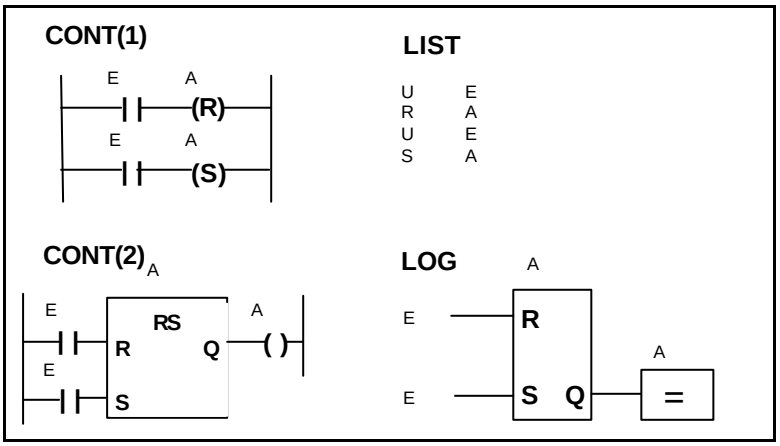



Les dernières instructions programmées sont traitées en priorité par la commande. Dans notre exemple, l'opération d'initialisation sera d'abord exécutée, puis la sortie A 2.0 sera de nouveau réinitialisée et restera pour tout le reste du traitement du programme dans cet état.

*Cette initialisation éphémère de la sortie est seulement effectuée dans l'image des processus. L'état du signal du module périphérique correspondant n'est ainsi pas influencé pendant le traitement du programme.*

### 2.9.2 INITIALISATION PRIORITAIRE

Conformément au paragraphe 4.10.1. la sortie A 2.1 est initialisée en priorité dans cet exemple.

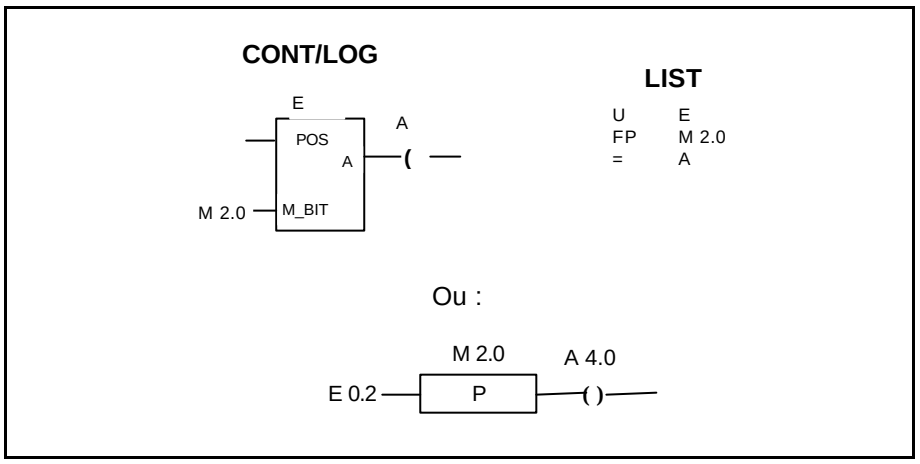


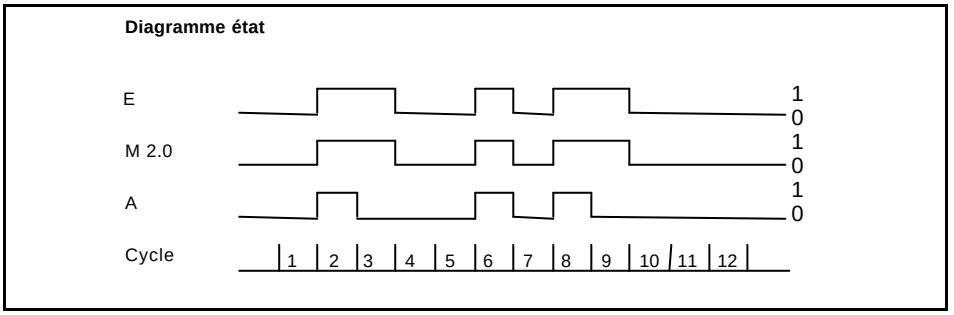
## 2.10 OPERATIONS SUR FRONT

Les opérations sur front détectent la *variation de niveau du signal* par exemple celle d'une entrée, contrairement aux opérations détectant un état du signal statique (0 ou 1). Le programme d'une opération sur front correspond à un contact détecteur de front dans un circuit relais.

### 2.10.1 FRONT MONTANT (FP)

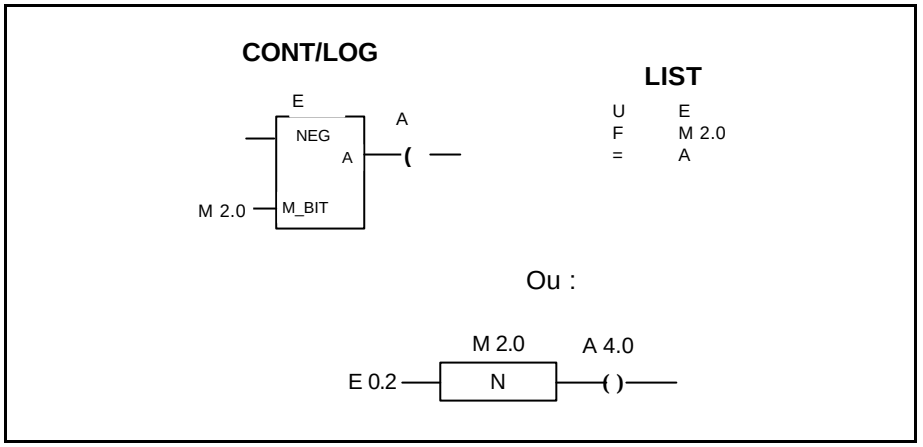
Si un front montant (pente positive) (Passage de '0'à '1') est reconnu sur E 0.2, alors A 4.0 est mise à 1 pour un cycle OB1. Cette sortie peut être réutilisée pour la mémoire interne, par exemple. Sur reconnaissance de front montant, le VKE, qui a conduit l'opération U, est enregistré dans la mémoire interne de front M 2.0, et est comparé avec le VKE du cycle précédent.
L'avantage de la deuxième catégorie de représentation en CONT/LOG réside dans le fait qu'il peut aussi y avoir des boîtes logiques à l'entrée des opérations sur front.

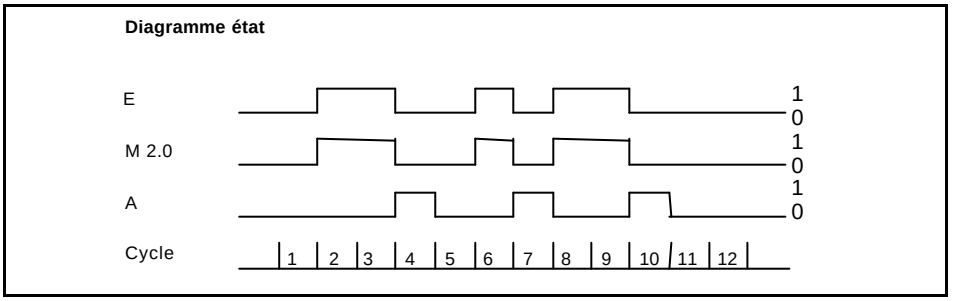


### 2.10.2 FRONT DESCENDANT (FN)

Si un front descendant (pente négative) (Passage de ‘1’à ‘0’) est reconnu sur E 0.2, alors A 4.0 est mise à 1 pour un cycle OB1. Cette sortie peut être réutilisée pour la mémoire interne, par exemple. Sur reconnaissance de front descendant, le VKE, qui a conduit l’opération U, est enregistré dans la mémoire interne de front M 2.0, et est comparé avec le VKE du cycle précédent.
L’avantage de la deuxième catégorie de représentation en CONT/LOG réside dans le fait qu’il peut aussi y avoir des boîtes logiques à l’entrée des opérations sur front.



## 2.11 FONCTIONS TEMPORELLES

Pour la réalisation de tâche de contrôles, on doit très souvent mettre en place diverses fonctions temporelles. Ces fonctions temporelles sont intégrées dans le module central de l'automate. On procède à la configuration des durées d'exécution et du démarrage de la fonction temporelle en passant par le programme utilisateur. Les automates SIMATIC mettent à votre disposition un certain nombre de variables temporelles ( dépendantes de la CPU ) avec différentes fonctions temporelles. A chaque variable temporelle est attribuée un mot de 16 bits.

Les fonctions suivantes peuvent être programmées pour une durée :

### 2.11.1 DEBLOQUER LE TEMPS (FR) SEULEMENT EN LIST

Un front montant ( de '0' à '1' ) sur le résultat logique de l'opération Déverrouillage (FR) débloque un temps.
Ce déverrouillage n'est pas nécessaire pour le démarrage ou pour les fonctions temporelles normales. Le déverrouillage est seulement employé pour un déclenchement à retardateur (trigger) c'est-à-dire pour débloquer l'écoulement de la durée. Cette remise en route de la durée est seulement possible si l'opération de démarrage travaille dans la suite avec VKE à 1.

***L'opération Déverrouillage (FR) est seulement disponible en LIST.***

### 2.11.2 DEMARRER LA DUREE (SI/SV/SE/SS/SA)

La variable temporelle est démarrée sur changement de signal ( front montant ) à l'entrée de démarrage. Pour démarrer une durée, servez-vous des trois instructions suivantes dans votre programme LIST :

- *Interrogation de l'état du signal*
- *Chargement d'une durée démarrage dans ACCU1*
- *Opération démarrage (au choix SI, SV, SE, SS ou SA)*

```
Ex. :
U    E 0.0
L    S5T#2S
SE   T5
```

### 2.11.3 PREDEFINITION DE LA VALEUR TEMPORELLE (TW)

Une variable temporelle doit toujours dépasser une certaine durée. La durée TW peut soit être renseignée par une constante fixe prédéfinie dans le programme, soit être définie comme mot d'entrée EW, comme mot de sortie AW, comme mot de données DBW/DIW, comme mot de données locales LW ou encore comme mot de mémoire interne MW. L'actualisation du temps décrémente la valeur temporelle d'une unité d'un intervalle qui est fixé comme base de temps.
Vous pouvez charger une valeur temporelle prédéfinie avec la syntaxe suivante :

- *L W#16#abcd*
  - avec : a = base de temps en binaire (c.-à-d. intervalle de temps, ou résolution, voir ci-dessous)
  - bcd = valeur temporelle en BCD
- *L S5T#aH_bbM_ccS_dddMS*
  - avec : a = Heures, bb = Minutes, cc = Secondes et ddd = Millisecondes
  - La base de temps est automatiquement choisie

**Base de temps :**

La base de temps définit l'intervalle dans lequel la valeur temporelle est décrémentée d'une unité. Les valeurs ne correspondant pas à un multiple exact de l'intervalle de temps seront tronquées. Les valeurs dont la résolution pour le domaine souhaité seront trop grandes seront arrondies.

| Temps | Binaire | Domaine temporel |
|---|---|---|
| 10ms | 00 | 10MS à 9S_990MS |
| 100ms | 01 | 100MS à 1M_39S_900MS |
| 1s | 10 | 1S à 16M_39S |
| 10s | 11 | 10S à 2H_46M_30S |

### 2.11.4 REINITIALISATION DE LA DUREE (R)

Un signal en entrée réinitialisation termine le traitement de la variable temporelle. La valeur temporelle actuelle est supprimée, et la sortie Q de la cellule temporelle est réinitialisée.

### 2.11.5 INTERROGER LA VALEUR TEMPORELLE (L/LC)

Une valeur temporelle est enregistrée dans un mot temporel en binaire. La valeur se trouvant dans le mot temporel peut être chargée dans l'ACCU comme nombre dual (DUAL) ou comme nombre BCD (DEC), et de là être transférée dans d'autres domaines d'opérandes.
En programmation LIST, vous avez le choix entre *L T1* en ce qui concerne l'interrogation du nombre dual et *LC T1* pour celle du nombre BCD.

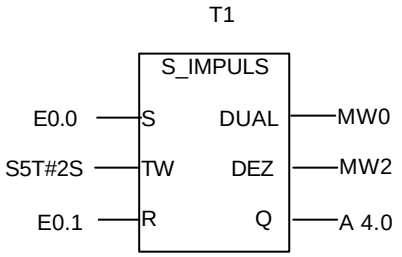
### 2.11.6 INTERROGER L'ETAT DU SIGNAL BINAIRE TEMPOREL (Q)

On peut interroger un temps sur son état de signal ('0' ou '1').
On peut utiliser les états du signaux avec U T1, UN T1, ON T1, etc…, comme d'habitude, et également les employer pour d'autres opérations logiques.
Vous pouvez choisir jusqu'à 5 types de durées différents :

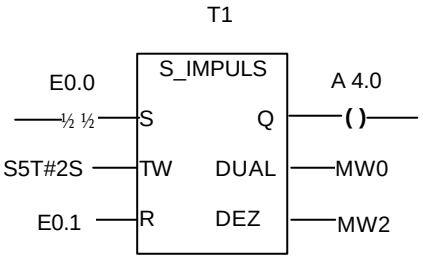
### 2.11.7 DUREE IMPULSION (SI)

La sortie d'une variable temporelle, qui est démarrée comme impulsion, délivre l'état de signal 1 après le démarrage (1). La sortie est réinitialisée quand la durée programmée est dépassée (2), si le signal de démarrage est réinitialisé à zéro (3) ou si l'entrée de réinitialisation du membre temporel présente l'état de signal 1 (4).
Sur front montant ( de '0' à '1' ) du résultat logique de l'opération de déverrouillage (FR) le temps redémarre (5). Cette reprise est seulement possible si l'opération de démarrage est toujours traitée avec le VKE à 1.

**LOG**



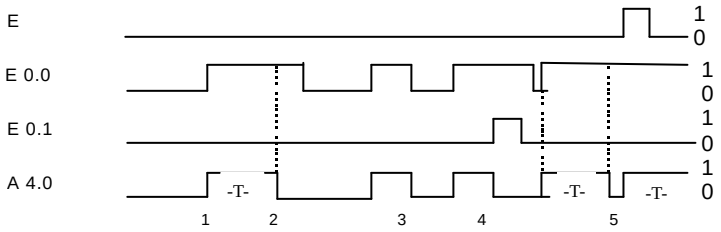

**CONT**




**LIST**

```
U    E 0.2
FR   T1      Déblocage du temps T1
             (LIST uniquement)
U    E 0.0
L    S5T#2S  Charge temps départ (2s) dans ACCU 1
SI   T1      Temps départ T1 en impulsion
U    E 0.1
R    T1      Réinitialisation de T1
L    T1      Charge temps T1 en binaire
T    MW0
LC   T1      Charge temps T1 en BCD
T    MW2
U    T1      Interrogation du temps T1
=    A 4.0
```

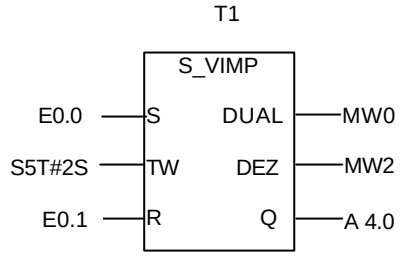
**Diagramme état signal avec déblocage**



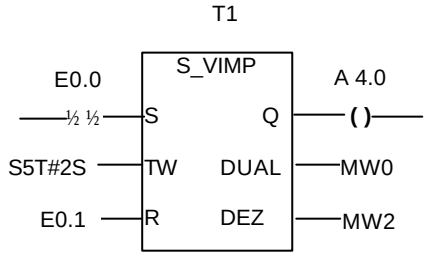
### 2.11.8 IMPULSION PROLONGEE (SV)

La sortie d'une variable temporelle, qui est démarrée comme impulsion prolongée, livre un état de signal 1 juste après le démarrage (1). La sortie sera réinitialisée, si la durée prédéfinie est écoulée (2) ou si l'entrée de réinitialisation de la fonction temporelle est connectée (5).
Une déconnexion de l'entrée de démarrage pendant que la durée s'écoule ne produit pas de réinitialisation de la sortie (commande maintenue) (3). Si un nouveau changement de signal à 1 s'effectue à l'entrée de démarrage pendant que la durée s'écoule encore, la variable temporelle est redémarrée ( post déclenchée ) (4).

**LOG**



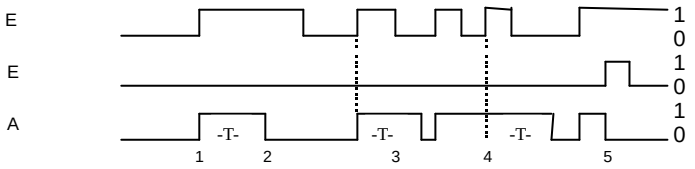

**CONT**




**LIST**

```
U    E 0.0
L    S5T#2S    Charge Temps départ (2s) dans  ACCU 1
SV   T1        Temps départ T1 en impulsion
U    E 0.1
R    T1        Réinitialise temps T1
L    T1        Charge Temps T1 en binaire
T    MW0
LC   T1        Charge Temps T1 en BCD
T    MW2
U    T1        Interrogation Temps T1
=    A 4.0
```

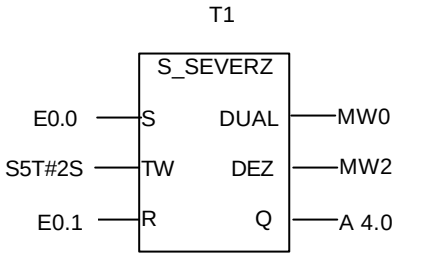
**Diagramme état**



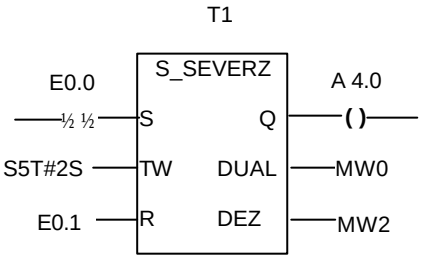
### 2.11.9 RETARD A L'ENCLENCHEMENT (SE)

La sortie d'une variable temporelle, qui est démarrée comme retard à l'enclenchement, livre un état de signal 1 juste après le démarrage, si la durée programmée est dépassée et que VKE présente encore 1 à l'entrée de démarrage (1). La connexion de l'entrée de démarrage a donc pour effet une connexion de la sortie Q retardée de la durée prédéfinie. La sortie sera réinitialisée, si la sortie est déconnectée (2) ou si l'entrée de réinitialisation de la variable temporelle présente l'état de signal 1 (3). La sortie Q ne sera pas connectée, si l'entrée de démarrage est déconnectée pendant que la durée s'écoule, ou si l'entrée de réinitialisation de la variable temporelle présente l'état de signal 1.

**LOG**



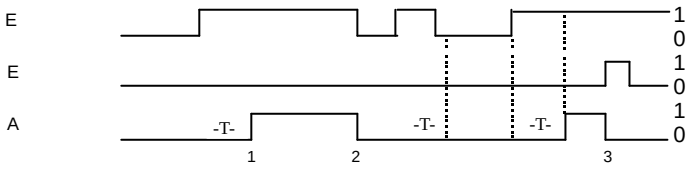

**CONT**




**LIST**

```
U    E 0.0
L    S5T#2S      Charge durée démarrage (2s) dans ACCU 1
SS   T1          Charge durée T1 en impulsion
U    E 0.1
R    T1          Réinitialise T1
L    T1          Charge durée T1 en binaire
T    MW0
LC   T1          Charge durée T1 en BCD
T    MW2
U    T1          Interrogation de la durée T1
=    A 4.0
```

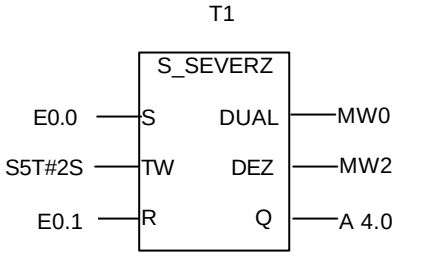
**Diagramme état**



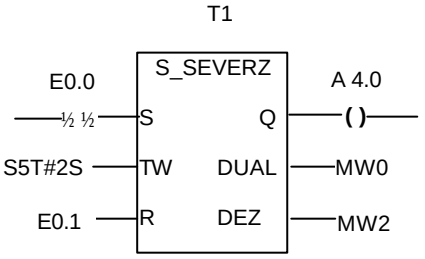
### 2.11.10 RETARD A L'ENCLENCHEMENT A MEMOIRE (SS)

La sortie d'une variable temporelle, qui est démarrée comme retard à l'enclenchement à mémoire, livre un état de signal 1 juste après le démarrage, si la durée programmée est dépassée (1). La fonction n'a plus besoin de VKE à 1 après le démarrage de l'entrée de démarrage, celle-ci peut donc être déconnectée (Commande maintenue) (3).
La sortie sera réinitialisée lorsque l'entrée de réinitialisation de la fonction temporelle sera connectée (2). Une déconnexion puis reconnexion de l'entrée de démarrage a pour effet un redémarrage de la fonction temporelle (post déclenchée), aussi longtemps que le temps s'écoule (4).

**LOG**



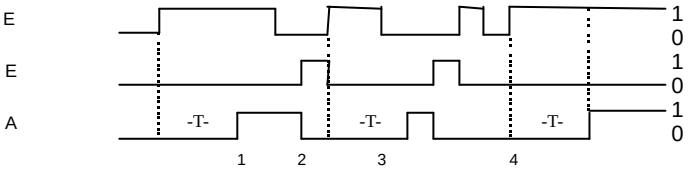

**CONT**




**LIST**

```
U    E 0.0
L    S5T#2S    Charge durée démarrage (2s) dans ACCU 1
SS   T1        Charge durée T1 en impulsion
U    E 0.1
R    T1        Réinitialise T1
L    T1        Charge durée T1 en binaire
T    MW0
LC   T1        Charge durée T1 en BCD
T    MW2
U    T1        Interrogation de la durée T1
=    A 4.0
```

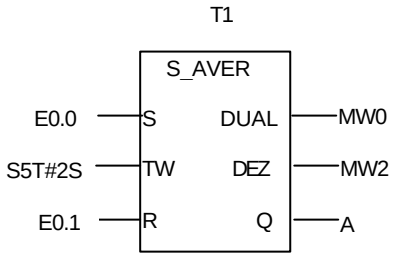
**Diagramme état**



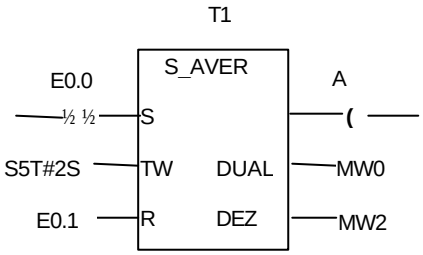
### 2.11.11 RETARD A LA COUPURE (SA)

Sur changement de signal ( en front montant ) à l'entrée de démarrage d'une variable temporelle, qui est démarrée en tant que retard à la coupure, la sortie Q de la fonction temporelle est connectée (1). Si l'entrée de démarrage est déconnectée, la sortie délivre l'état de signal 1 jusqu'à ce que le temps programmé soit dépassé (2). La déconnexion de l'entrée de démarrage (front descendant) produit donc une déconnexion retardée de la durée prédéfinie de la sortie. La sortie de la variable temporelle est aussi déconnectée, si l'entrée de réinitialisation délivre l'état de signal 1 (4). Un redémarrage de la fonction temporelle provoque, pendant que la durée s'écoule, l'immobilisation du temps courant et le redémarrage pour la prochaine déconnexion de l'entrée de démarrage (3).
Si l'entrée de démarrage et l'entrée de réinitialisation de la fonction temporelle délivrent toutes deux l'état de signal 1, la sortie de la variable temporelle est initialisée si la réinitialisation dominante est connectée (5).

**LOG**



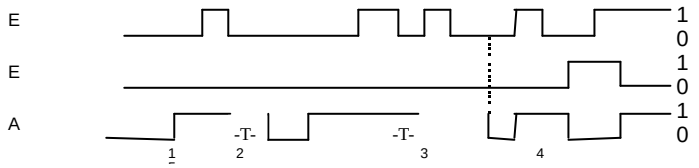

**CONT**




**LIST**

```
U   E 0.0
L   S5T#2S    Charge durée départ (2s) dans ACCU
S   T1        Durée départ T1 en
U   E 0.1
R   T1        Réinitialisation de
L   T1         Charge durée T1 en
T   MW0
LC  T1        Charge  T1 en BCD
T   MW2
U   T1        Interrogation durée
=   A
```

**Diagramme état**



## 2.12 GENERATEUR D'HORLOGE

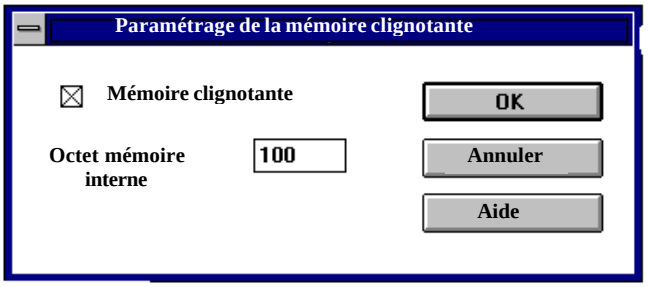
Les générateurs d'horloge sont mis en place pour différentes tâches de contrôle, de surveillance et de commande. En technique du numérique, on les désigne en tant que bascule bistable astable.
On a souvent besoin dans la pratique d'une fréquence clignotante pour les messages de fonctionnement ou de dérangement.

*Il existe une mémoire clignotante paramétrable dans la CPU S7-300, qui peut être configurée avec l'outil matériel.*

**Paramétrer la mémoire clignotante :**

Les mémoires clignotantes sont des mémoires internes à l'intérieur d'un octet de mémoire clignotante. Pour faire d'un octet mémoire interne quelconque de la CPU un octet mémoire clignotante, on double-clique sur la ligne CPU dans l'outil de configuration. Une mémoire clignotante change périodiquement sa valeur binaire.

Si vous activez la mémoire clignotante ( la case est cochée), alors vous devez aussi déterminer le numéro de l'octet de la mémoire interne. On ne peut pas utiliser l'octet de mémoire interne pour la sauvegarde intermédiaire de données.




**Période d'horloge :**

Chaque bit de l'octet de la mémoire clignotante est attribué à une période/fréquence. L'attribution est réalisée de la manière suivante :

| **Bit :** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| Période (s) : | 2 | 1,6 | 1 | 0,8 | 0,5 | 0,4 | 0,2 | 0,1 |
| Fréquence (Hz) : | 0,5 | 0,625 | 1 | 1,25 | 2 | 2,5 | 5 | 10 |

## 2.13 OPERATIONS DE COMPTAGE

En technique des régulations, on a besoin de fonctions de comptage pour déterminer le nombre d'entités ou d'impulsions, pour évaluer les durées et les distances. En SIMATIC S7 les compteurs sont déjà intégrés dans les modules centraux. Ces compteurs possèdent un espace mémoire propre. Le domaine des valeurs que peut prendre le compteur va de 0 à 999.

On peut programmer pour un compteur les fonctions suivantes :

### 2.13.1 DEBLOQUER COMPTEUR (FR) SEULEMENT EN LIST

Un front montant ( de '0' à '1' ) sur résultat logique de l'opération Libérer (FR) débloque le compteur. Un déblocage de compteur n'est nécessaire ni pour l'initialisation d'un compteur, ni pour des opérations normales. Si l'on souhaite toutefois initialiser un compteur, l'incrémenter ou le décrémenter sans pour autant avoir un front montant avant l'opération compteur ( ZV, ZR ou S ), alors on peut effectuer un déblocage. Cela n'est cependant seulement possible que si le bit VKE a l'état de signal 1 avant l'opération correspondante ( ZV, ZR ou S ).

***L'opération Libérer (FR) est seulement disponible dans le mode de représentation LIST.***

### 2.13.2 INCREMENTATION (ZV)

La valeur du compteur désigné est incrémentée par pas de 1. La fonction sera active seulement sur front montant avant l'opération logique programmée ZV. Si le compteur atteint la borne maximale 999, il ne s'incrémentera plus. *(La transmission ne pourra plus avoir lieu ! )*

### 2.13.3 DECREMENTATION (ZR)

La valeur du compteur désigné est décrémentée par pas de 1. La fonction sera active seulement sur front montant avant l'opération logique programmée ZR. Si le compteur atteint la borne minimale de 0, il ne se décrémentera plus. *( Le compteur ne travaille qu'en valeurs positives ! )*

### 2.13.4 INTIALISER COMPTEUR (S)

Pour initialiser un compteur, saisissez 3 instructions dans votre programme LIST :

- Interrogation d'un état du signal
- Chargement d'une valeur de compteur
- Mise du compteur à la valeur chargée
  Cette fonction sera seulement traitée sur front montant de l'interrogation.

| Par ex. : | |
|---|---|
| U | E 2.3 |
| L | C#5 |
| S | Z1 |

### 2.13.5 REQUETE VALEUR COMPTEUR (ZW)

Quand un compteur est initialisé, le contenu de l'ACCU 1 est employé comme valeur de comptage. Vous avez la possibilité de charger la valeur compteur soit en binaire soit en BCD.
Les opérandes suivants sont possibles :

- *Mot entrée* *EW ..*
- *Mot sortie* *AW ..*
- *Mot mémoire* *MW ..*
- *Mot données* *DBW/DIW ..*
- *Mot données locales* *LW ..*
- *Constantes* *C#5, 2#...etc.*

### 2.13.6 REINITIALISATION COMPTEUR (R)

Si VKE est à 1, le compteur est mis à 0 (réinitialisation). Si VKE est à 0, le compteur reste dans son état. La réinitialisation d'un compteur agit statiquement. Après une réinitialisation en bonne et due forme, le compteur ne peut ni être initialisé, ni être incrémenté/décrémenté.

### 2.13.7 INTERROGER VALEUR COMPTEUR (L/LC)

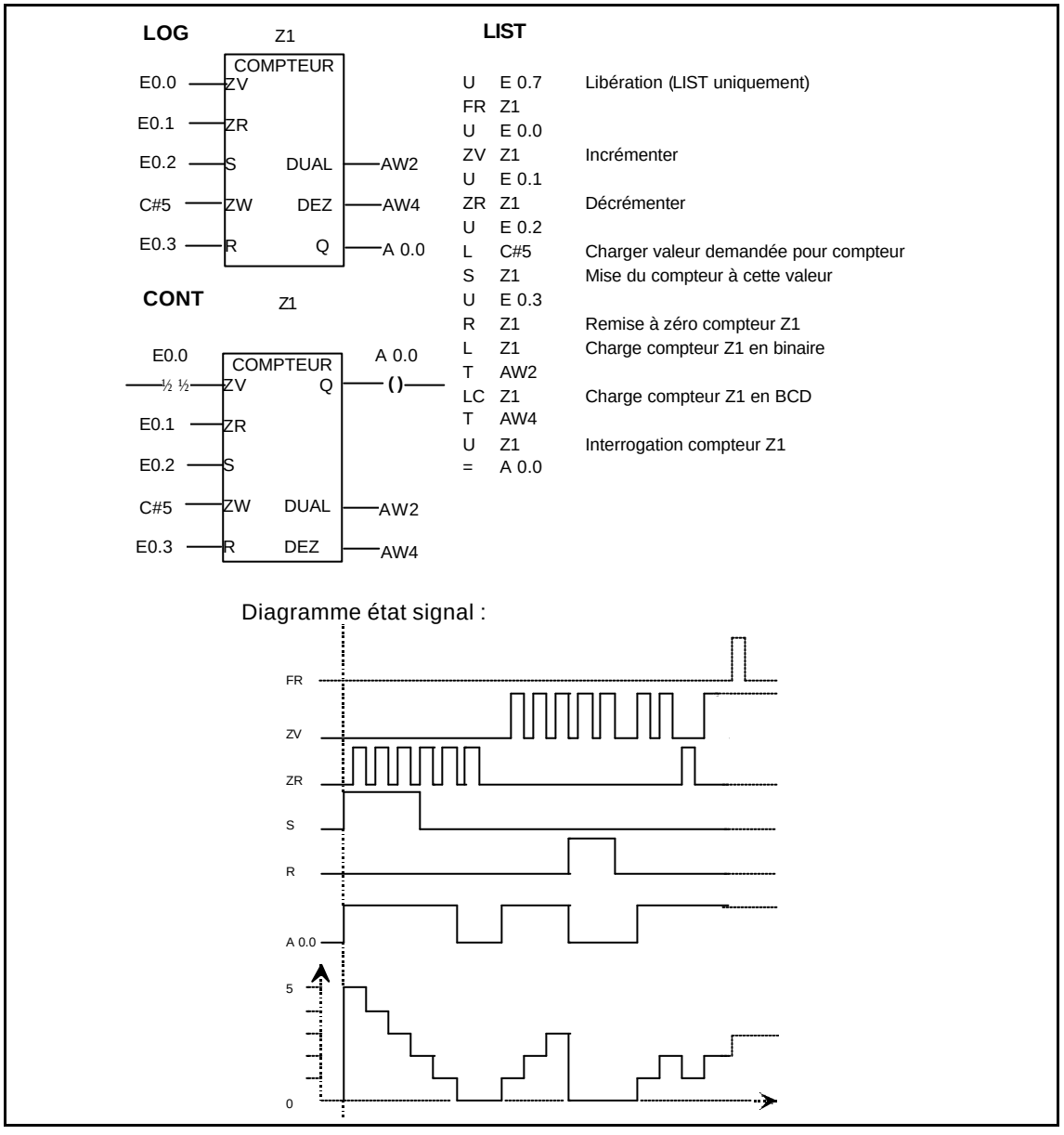
Une valeur compteur est stockée dans un mot compteur en binaire. La valeur se trouvant dans le mot temporel peut être chargée dans l'ACCU comme nombre dual (DUAL) ou comme nombre BCD (DEC), et de là être transférée dans d'autres domaines d'opérandes.
En programmation LIST, vous avez le choix entre *L Z1* en ce qui concerne l'interrogation du nombre dual et *LC T1* pour celle du nombre BCD.

## 2.13.8 INTERROGER L'ETAT SIGNAL DU COMPTEUR EN BINAIRE (Q)

On peut interroger le compteur sur son état de signal. La réponse correspondra aux cas suivants :

*Etat signal 0 = Le compteur est à 0;*
*Etat signal 1 = Le compteur travaille, c.-à-d. il est prêt à compter.*

On peut utiliser les états du signaux avec *U Z1, UN Z1, ON Z1*, etc..., comme d'habitude, et également les employer pour d'autres opérations logiques.

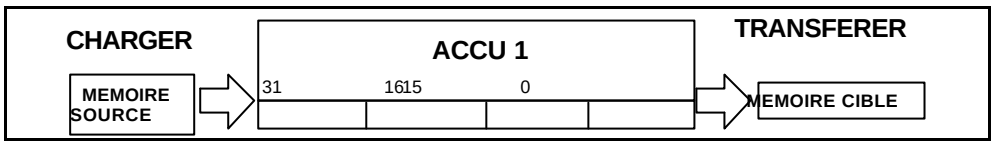


## 2.14 OPERATIONS DE CHARGEMENT ET DE TRANSFERT (L/T)

En langage STEP7, les opérations de chargement et de transfert permettent l'échange d'information octet/mot/double mot entre les modules d'entrée et de sortie, entre les images des processus des entrées et des sorties, entre les mémoires temps/compteur/clignotantes, ainsi qu'entre les blocs de données.
Cet échange d'information se s'effectue pas directement, il s'effectue toujours via l'accumulateur 1 (ACCU 1).
L'ACCU 1 est un registre dans le processeur et sert de mémoire intermédiaire.

Le flux d'information est directionnel :

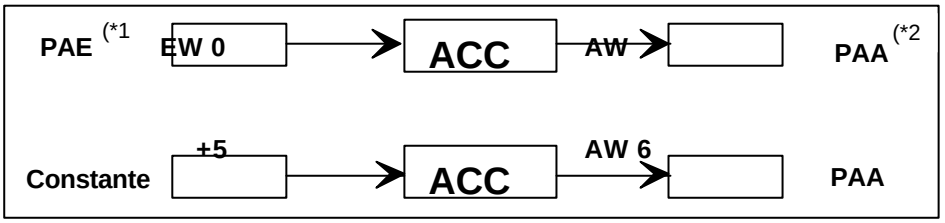
*CHARGER : de la mémoire source à l'ACCU 1*
*TRANSFERER : de l'ACCU à la mémoire cible*




Lors du chargement, le contenu de la mémoire source désignée est copié et écrit dans l'ACCU 1. Le contenu s'y trouvant jusque là est transmis à l'ACCU 2.
Lors du transfert, le contenu de l'ACCU 1 est copié et écrit dans la mémoire cible désignée.
Puisque le contenu de l'accumulateur a été seulement copié (et pas *déplacé*) il reste à disposition pour d'autres opérations de transfert.

**LIST**

```
: L   EW    0
: T   AW    4
: L   +5
: T   AW    6
.
```




*1: Image des processus des entrées *2: Image des processus des sorties

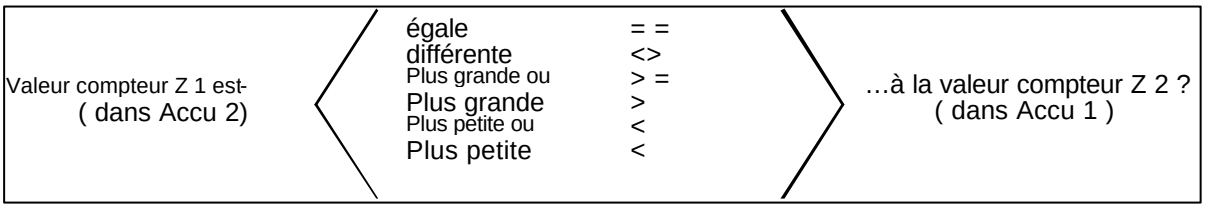
Charger et transférer sont deux opérations inconditionnelles, complètement indépendantes du résultat logique et exécutées à chaque cycle.

## 2.15 FONCTIONS DE COMPARAISON

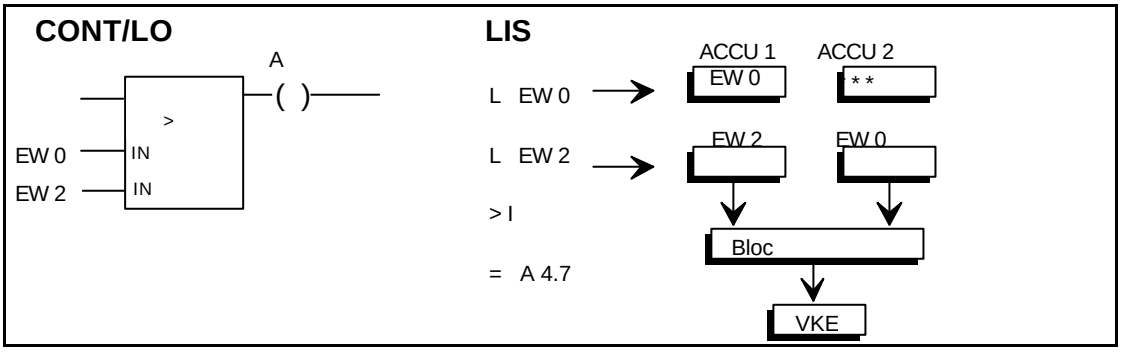
Le langage de programmation STEP7 vous offre la possibilité de comparer 2 valeurs compteur directement et de propager immédiatement le résultat de la comparaison ( VKE ). La condition préalable pour se faire, est que les deux nombres aient le même format.
Les couples de valeurs suivants peuvent être comparés :

- *2 entiers ( 16 bits* *Symbole : I )*
- *2 entiers ( 32 bits* *Symbole : D )*
- *2 réels ( virgule flottante, 32 bits,* *Symbole : R )*

Vous pouvez choisir jusqu'à 6 types différents d'opérateurs de comparaison :




A l'aide des fonctions de comparaison, on compare directement deux valeurs se trouvant l'une dans l'ACCU1 et l'autre dans l'ACCU2. La première opération de chargement charge le premier opérande dans l'ACCU1. La deuxième opération de chargement transfert tout d'abord le premier opérande depuis l'ACCU1 à l'ACCU2, puis charge le deuxième opérande (par ex. EW 2) dans l'ACCU1. Par la suite, les deux valeurs se trouvant dans les deux accumulateurs sont comparées bit à bit dans un bloc arithmétique. Le résultat de la comparaison est binaire. Si la comparaison est vraie, on obtient le résultat logique 1. Si elle est fausse, VKE est mis à 0.



## 2.16 ORGANISATION DU PROGRAMME

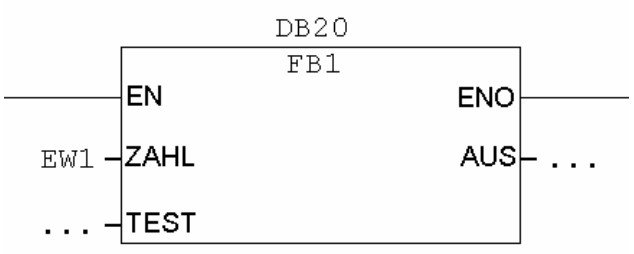
### 2.16.1 APPEL DE BLOC (CALL)

Avec l'appel de bloc *CALL* vous pouvez appeler des fonctions (FC) et des blocs de fonctions (FB) ainsi que des fonctions système (SFC) et des blocs de fonctions système (SFB). On peut simultanément transférer des paramètres, écrire des variables, ouvrir les blocs locaux correspondants de FB ou SFB. S'il n'y a pas de variables déclarées dans le module appelé, alors cette commande est équivalente à la commande UC.

**LIST**

| | | |
|---|---|---|
| CALL | FB1, DB20 | |
| ZAHL := | EW 1 | ZAHL (Paramètre formel) est attribué à EW 1 (paramètre actif). |
| AUS := | | AUS (Paramètre formel) n'est attribué à aucun paramètre. |
| TEST := | | TEST (Paramètre formel) n'est attribué à aucun paramètre. |

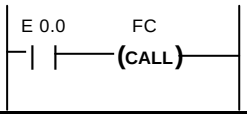
**CONT/LOG**




### 2.16.2 APPEL DE BLOC CONDITIONNEL (CC)

A l'aide de l'appel de bloc CC, vous pouvez appeler des fonctions (FC) et des blocs de fonctions (FB) ainsi que des fonctions système (SFC) et des blocs de fonctions système (SFB). Vous ne pouvez toutefois pas transférer des paramètres ou écrire des variables.
L'appel ne sera exécuté que si le résultat logique vaut 1.

**CONT/LOG**



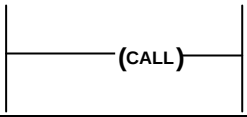

**LIST**

```
U     E
C     FC
```

### 2.16.3 APPEL DE BLOC INCONDITIONNEL (UC)

A l'aide de l'appel de bloc *UC*, vous pouvez appeler des fonctions (FC) et des blocs de fonctions (FB) ainsi que des fonctions système (SFC) et des blocs de fonctions système (SFB). Vous ne pouvez toutefois pas transférer des paramètres ou écrire des variables.
L'exécution de l'appel est indépendante du résultat logique.

**CONT/LOG**



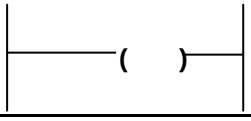

**LIST**

```
U     FC
```

### 2.16.4 OUVRIR UN BLOC DE DONNEES (AUF)

A l'aide de l'opération Ouvrir un Bloc de Données (AUF), vous pouvez ouvrir un bloc de données (DB) ou un bloc instance de données (DI), afin de pouvoir accéder aux données qui y sont contenues ( par ex. avec les opérations de chargement et de transfert ).

**CONT/LOG**




**LIST**

```
AUF   DB
L     DBW 0
T     MW 1
```

### 2.16.5 TERMINAISON CONDITIONNELLE DE BLOC (BEB) SEULEMENT EN LIST

Selon le résultat logique, cette opération termine le traitement du bloc en cours et revient au bloc qui a lancé la demande de terminaison.
Cette opération s'effectue seulement si le résultat logique vaut '1'.

**LIST**

```
U     E
BEB
```

### 2.16.6 TERMINAISON INCONDITIONNELLE DE BLOC (BEA) SEULEMENT EN LIST

Cette opération termine le traitement du bloc en cours et revient au bloc qui a lancé la demande de terminaison.
L'exécution de cette opération est indépendante du résultat logique.

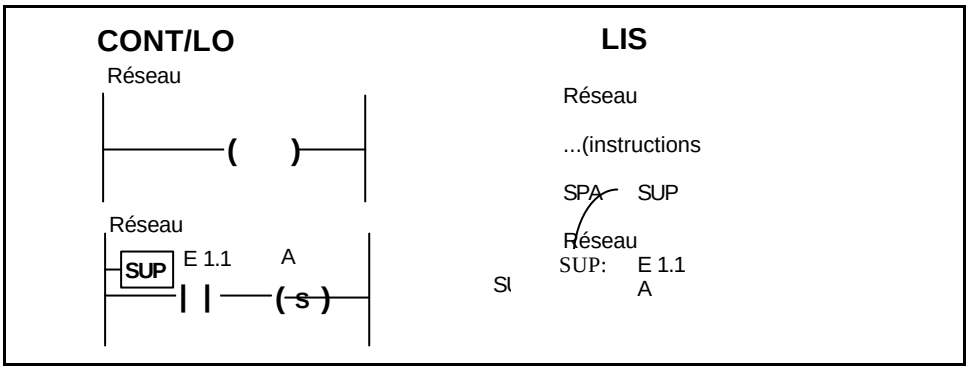
**LIST**

```
...(instructions
BEA
```

## 2.17 OPERATIONS DE SAUTS

### 2.17.1 SAUT INCONDITIONNEL (SPA)

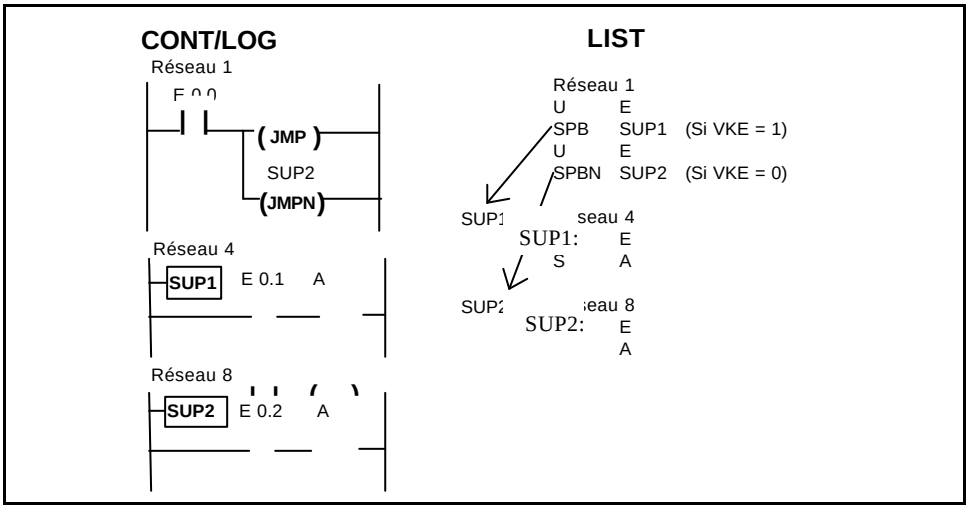
L'opération SPA interrompt le déroulement normal du programme et saute à l'index de saut indiqué dans l'opérande.
Le saut s'effectue quel que soit le résultat logique.




### 2.17.2 SAUT CONDITIONNEL (SPB/SPBN)

Les opérations de saut conditionnel interrompent le déroulement normal du programme et procèdent à un saut à l'index de saut indiqué dans l'opérande.
Le saut s'effectue en fonction du contenu du résultat logique.
Vous pouvez exécuter les opérations de saut conditionnel suivantes :

- *SPB : saut si VKE = 1*
- *SPBN : saut si VKE = 0*



### 2.17.3 BOUCLE DE PROGRAMME (LOOP) SEULEMENT EN LIST

A l'aide d'une boucle de programme (LOOP), vous pouvez traiter un bout de programme plusieurs fois à la suite.
Pour cela vous devez charger une constante dans le mot de poids faible de l'ACCU1. Ce nombre sera ensuite décrémenté de 1 par l'opération LOOP. Ensuite cette valeur compteur sera comparée à <>0. Si cela ne vaut pas '0', alors l'opération LOOP conduit à un saut à l'index (‚NEXT' dans notre exemple) ; sinon l'instruction suivante sera exécutée.

```
        L     5
NEXT: T     MW 10
        |
        |
        L     MW 10
        LOOP  NEXT
```

***La boucle de programme (LOOP) n'est disponible qu'en mode de représentation LIST.***

## 2.18 OPERATIONS NULL

### 2.18.1 OPERATION NULL 0 / 1 (NOP0/NOP1) SEULEMENT EN LIST

Ces opérations n'exécutent aucune fonction et n'ont pas d'incidence sur le mot de statut. L'interpréteur a besoin des opérations NULL pour l'interprétation retour, par ex. de LIST vers CONT.

## 2.19 TRAITEMENT DU VKE

En STEP7, on trouve des opérations qui peuvent changer la valeur du résultat logique (VKE) Puisque dans celles-ci VKE est directement modifié, ces opérations n'ont pas d'opérandes.

### 2.19.1 NEGATION DU VKE (NO) SEULEMENT EN LIST

Vous pouvez effectuer dans votre programme une négation du VKE actuel à l'aide de l'opération NON (NOT) (négation). Si le VKE actuel est à 0, alors l'opération NON le change en 1, si le VKE actuel est à 1, alors l'opération NON le change en 0.

### 2.19.2 METTRE A 1 LE VKE (SET) SEULEMENT EN LIST

Vous pouvez mettre le bit VKE inconditionnellement à 1 dans votre programme en utilisant l'opération MISE A 1 (SET).

### 2.19.3 REMISE A ZERO DU VKE (CLR) SEULEMENT EN LIST

Vous pouvez remettre le bit VKE inconditionnellement à 0 dans votre programme en utilisant l'opération REMISE A 0 (CLR).

### 2.19.4 SAUVEGARDE DU VKE (SAVE) SEULEMENT EN LIST

Vous avez la possibilité de sauvegarder le VKE dans le bit de statut (BIE) du mot de statut dans votre programme à l'aide de l'opération SAUVER (SAVE), pour pouvoir vous en servir ultérieurement.

Le mot de statut contient des bits auxquels peuvent accéder en opérandes les opérations logiques mot/bit.

| | BIE | A1 | A0 | OV | OS | OR | STA | VKE | /ER |
|---|---|---|---|---|---|---|---|---|---|
| Par ex.: | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |

| **Liste d'instructions :** | **Etat signal :** | **Résultat logique (VKE):** |
|---|---|---|
| SET | | 1 |
| = M 1.0 | 1 | |
| = E 0.0 | 1 | |
| CLR | | 0 |
| = M 1.0 | 0 | |
| = E 0.0 | 0 | |
| NOT | 1 | |
| SAVE | 1 | sauvegarde dans le bit BIE du mot de statut |