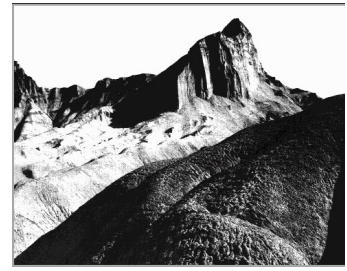
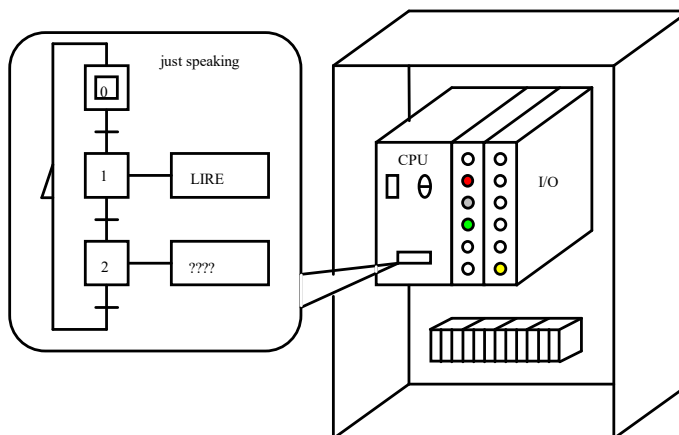


Lycée du Pré Saint Sauveur
39200 Saint Claude



GRAF CET

Méthode d'implémentation du GRAFCET
sur automate à instruction de type list.
Application sur SIEMENS CPU 212 et CPU 214



9 fevrier 98

1996-98 **Y. - Cordier**
J.L.- Ridacker

La société SIEMENS commercialise un nouvel automate d'entrée de gamme (a bien y regarder ce n'est pas vraiment de l'entrée car les possibilités logicielles et matériels sont très étendues). A leur habitude, cette société n'est pas très friante du sacrosaint GRAFCET. Suite à une étude menée en 96 sur les CPU103 et 95, voici une étude de même facture (plus simple !) sur l'implémentation du langage GRAFCET sur CPU102 et 104.

SOMMAIRE

- I Présentation générale.*
- II Notions préliminaires*
 - 2.1 Notion de table*
 - 2.2 Opérations sur les tables*
 - 2.3 Les règles du GRAFCET*
 - 2.4 Le cycle classique d'un automate*
 - 2.5 Travail du programmeur*
 - 2.6 Chronologie des actions*
- III Possibilités et tables*
 - 3.1 Possibilités*
 - 3.2 Les tables*
 - 3.3 Utilisation des tables*
 - 3.4 Adresses particulières*
- IV Architecture logicielle*
 - 4.1 possibilités*
 - 4.2 Architecture proposée*
 - 4.3 Utilisation des bits système*
- V Détail des unités logicielle*
 - 5.0 OB1*
 - 5.1 SBR 0*
 - 5.2 SBR 1*
 - 5.3 SBR 2*
 - 5.4 SBR 3*
 - 5.5 SBR 4*
 - 5.6 SBR 5*
 - 5.7 SBR 6*
 - 5.8 SBR 8*
 - 5.9 SBR 10*
- VI Taille mémoire*

Suivi de trois exemples commentés de complexité croissante.

Auteur : Cordier Yves

Professeur de (on ne sait plus trop bien au juste)
Lycée du Pré Saint Sauveur 39200 Saint Claude



0 SOMMAIRE

<i>I Présentation générale.</i>	P1
<i>II Notions préliminaires</i>	P2
2.1 Notion de table	P2
2.2 Opérations sur les tables	P3
2.3 Les règles du GRAFCET	P3
2.4 Le cycle classique d'un automate	P4
2.5 Travail du programmeur	P4
2.6 Chronologie des actions	P5
<i>III Possibilités et tables</i>	P6
3.1 Possibilités	P6
3.2 Les tables	P6
3.3 Utilisation des tables	P7
3.4 Adresses particulières	P7
<i>IV Architecture logicielle</i>	P8
4.1 possibilités	P8
4.2 Architecture proposée	P8
4.3 Utilisation des bits système	P10
<i>V Détail des unités logicielle</i>	P10
5.0 OBI	P10
5.1 SBR 0	P11
5.2 SBR 1	P11
5.3 SBR 2	P12
5.4 SBR 3	P12
5.5 SBR 4	P13
5.6 SBR 5	P14
5.7 SBR 6	P15
5.8 SBR 8	P15
5.9 SBR 10	P16
<i>VI Taille mémoire</i>	P16

I Présentation générale.

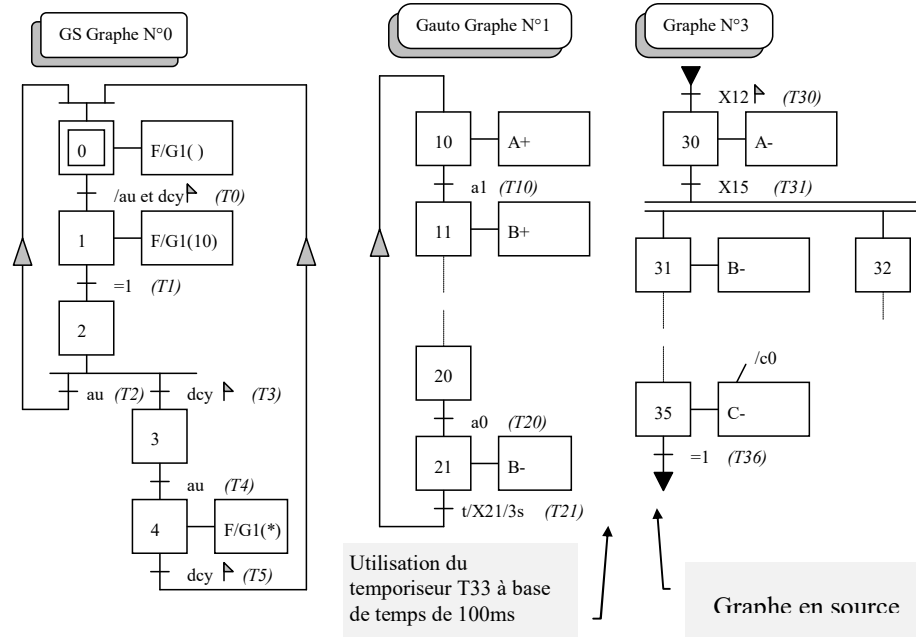
Les automates programmables du commerce peuvent grossièrement se diviser en deux catégories : ceux ayant un noyau GRAFCET (Ex TSX OMRON ..) et ceux ayant un noyau orienté manipulation de bits mots (Ex : Siemens ABB ...). L'implémentations du langage de description séquentiel d'automatisme GRAFCET sur cette dernière catégorie d'automate relève souvent de recettes de cuisine. Ces recettes fonctionnent relativement bien dans les cas simples, mais, posent des problèmes de non respect des 5 (CINQ!) règles du GRAFCET dans beaucoup de cas. De plus la relecture d'un programme pour modification n'est pas chose aisée.

Durant ces pages, les exemples seront traités en utilisant soit le mnémonique des automates Siemens, série 7 S7 200 soit un macro-langage issu de l'analyse informatique. Il est possible d'adapter ces exemples en un autre langage, je vous en laisse le soin.



A partir du chapitre V, le listing correspond au squelette (valable quelque soit le système de graphes) de programmation de la méthode proposée.

Pour la partie programme dépendant de l'application, nous donnerons des exemples en s'appuyant sur les morceaux de graphes ci-contre.



Comme ce système de graphe ne correspond à aucunes Parties Opératives réelles, que les puristes me pardonnent, l'amalgame très (trop) flagrant entre Graphe du point de vue P.O. et graphe du point de vue P.C. sur les morceaux de code exemple.

II Notions préliminaires

2.1 Notion de table

Durant toute cette présentation, nous ferons appel à des tables de bits représentant les activités d'étapes et ou tout autres informations relatives à ces étapes. Dans un automate basé sur un microprocesseur, l'unité de stockage mémoire est l'octet le mot (2 octets) ou le double mot (4 octets), ce qui donne par table 8, 16 ou 32 informations binaires que l'on peut manipuler indépendamment ou par paquet.

Exemple de table sur un automate siemens

Nom d'étape	0	1	.	7	10	11	.	17		37
Variable d'activité associée	X0	X1	.	X7	X10	X11	.	X17		X37
Variable automate VB10, VB11, VB12, VB13	V10.0	V10.1		V10.7	V11.0	V11.1	.	V11.7		V13.7

Par la suite nous parlerons des tables d'activité d'étape T_p et T_s . Dans les exemples, pour plus de clarté, nous utiliserons indistinctement les adressages automates ($Vx.y$) ou les variables d'activité associées (Xi). De même pour les transitions, nous utiliserons les pseudo adressages t_i .

Pour augmenter la lisibilité des programmes, pour calquer au mieux les octets et mots, les $Xn8$ et $Xn9$ ne seront pas utilisés dans la description au sens automate.



2.2 Opérations sur les tables

Toutes les opérations booléennes classiques ET (.), OU (+), NON (/), OU exclusif (\oplus) bit à bit s'appliquent aussi sur les tables. Ecrire $VB30=VB10$ et $VB0$ revient à écrire pour chaque information booléenne de la table $VB30.x=VB10.x$ et $VB20.x$

Remarque : - Certains automates ne sont pas équipés d'une instruction de négation de mots (négation bit à bit). Il est possible de contourner ce problème en utilisant l'instruction de Ou exclusif. En effet il est facile de montrer que $\bar{T} = T \oplus FFFF$ ou $FFFF$ représente un mot 16 bits tous à 1 si T est une table de 16 bits ($\bar{\bar{T}} = T \oplus FFFFFFFF$ ou $FFFFFFFF$ représente un mot 32 bits tous à 1 si T est une table de 32 bits.

- l'automate S7 212 et 214 possède une instruction de complément de mot $INV<mot>$.

2.3 Les règles du GRAFCET

Règle N°1

Les étapes initiales, repérées par un double carré, sont activées inconditionnellement à l'initialisation de l'automatisme, au début du fonctionnement.

Règle N°2

Le franchissement d'une transition ne peut se produire que si la transition est validée (étapes précédentes actives) et si la réceptivité associée à la transition est vraie.

Si les deux conditions sont réunies, la transition devient franchissable et est alors obligatoirement franchie.

Règle N°3

Le franchissement d'une transition entraîne en même temps l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

Règle N°4

Plusieurs transitions simultanément franchissables sont simultanément franchies.

Règle N°5

Si, au cours du fonctionnement une étape doit être à la fois désactivée et activée, elle reste active.

Commentaires

A l'initialisation (mise sous énergie) d'un système les seules étapes actives sont les étapes initiales. Elles sont supposées actives depuis toujours et donc, le front montant de l'activité de ces étapes est faux au démarrage du système (CF. chap. 4.2 P8, 5.0 P10, 5.1 P11).

A l'initialisation, l'état de toutes les variables internes (et externes) est supposé stable depuis longtemps. Tous les états résultant de fronts montants et descendants sont faux (CF. chap. 4.2 P8, 5.0 P10, 5.1 P11).



Un graphe figé peut évoluer par forçage.

Une étape d'un graphe peut être à la fois forcée à 0 par un graphe et à 1 par un autre graphe de 'niveau' supérieur, elle est alors forcée à 1.

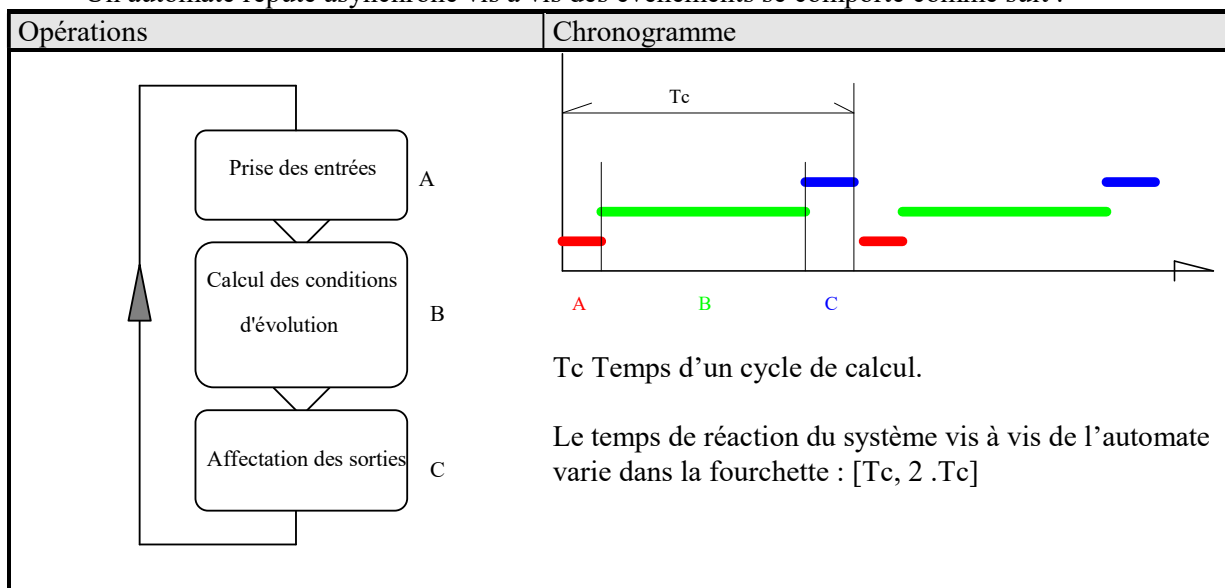
Les graphes sont hiérarchisés et un graphe ne peut être forcé que par un graphe de niveau supérieur.

Les systèmes de graphes ou un graphe s'auto-force et ou un graphe N°1 force un graphe N°2 qui en même temps force un graphe N°1 ne sont pas valide. On pourrait dans de tels cas assister à un verrouillage temporel définitif du système de graphe.

Une configuration stable d'état des étapes du système de graphe le reste suffisamment longtemps pour effectuer les actions correspondantes.

2.4 Le cycle classique d'un automate

Un automate réputé asynchrone vis à vis des événements se comporte comme suit :

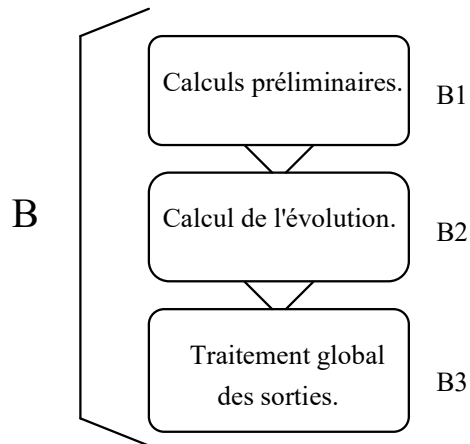


2.5 Travail du programmeur

Le programmeur n'a accès qu'à la case B. Afin d'éviter les problèmes de réaffectation des sorties, la méthode à utiliser pour la programmation doit avoir une certaine rigueur et se présenter comme suit :



Détail de la partie B



B1 : Dans cette partie on traite :
- Les fronts
- Les conversions analogiques numériques

B2 : Le traitement du corps du GRAFCET et ou toute autre méthode de description.

B3 : Différentes opérations sont à effectuer dans l'ordre pour un bon fonctionnement :
- Traitement des forçages
- Traitement des compteurs
- Lancement des temporisations
- Traitement global des sorties et figeages
- *Conversions numériques analogiques.*

soit la chronologie suivante :

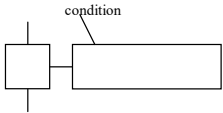
Traitement préliminaire
Traitement des graphes
Traitement des forçages
Traitement des compteurs
Lancement des temporisations
Traitement des affectations des sorties et figeages

Nous reviendrons ultérieurement plus en détail sur le pourquoi et l'ordre impératif des actions de B3.

2.6 Chronologie des actions

Cette chronologie est imposée par le fonctionnement séquentiel des instructions et la présence de certaine actions vis à vis des autre.



<i>Remarque</i>	<i>Conséquence</i>
Les actions associés à une étape correspondent à un état stable (au moins un tour de calcul) de l'étape	Tous les traitements d'évolution doivent être effectués avant le combinatoire de sortie.
Les état instables des sorties peuvent nuire au fonctionnement de la machine automatisée.	Les forçages doivent être traitées avant le combinatoire de sortie.
Le traitement d'actions  dans lequel la condition est un résultat de compteur ou une temporisation oblige de connaître l'état 'stable' de la condition avant le traitement des sorties	Le traitement des temporisateurs et compteurs doit se faire avant le combinatoire de sortie.
Un graphe figé peut évoluer par forçage.	Les forçages sont des actions prioritaires aux figeages et donc seront traités avant les figeages.
Les actions de figeages correspondent à un verrouillage de l'état actuel des variables d'étape au tour de calcul suivant .	Les figeages sont des actions qui peuvent être traitées dès que l'état 'stable' des variables d'étapes est déterminé soit n'importe ou après le traitement des forçages. Un figeage de graphe correspond en fait à une action (semblable à une action associée à une étape) qui bloque l'évolution des transition. Ce blocage peut être réalisé par dévalidation systématique des receptivités des transitions des graphes figés. Nous le traiterons ces opérations dans le combinatoire de sortie.

III Possibilités et tables

3.1 Possibilités

Nous proposerons ici une structure logicielle permettant d'implémenter les cinq règles du grafcet sur une CPU 214 avec les possibilités suivantes:

- 16 graphes
- 160 transitions réparties sur les divers graphes
- 80 étapes

3.2 Les tables



Nom	Rôle	Taille	Adresse	Exemple
T _{fig}	Table de figeage des graphes	16 bits	VB0..VB1= VW0	V0.1
T _p	Etat des étapes au tour précédent	80 bits	VB10..VB19 = VW10+VW12+VW14+VW16 +VW18	V11.1 = X10 _{prec}
T _s	Etat des étapes a prendre en compte en fin de tour de calcul	80 bits	VB20..VB29 = VW20+VW22+VW24+VW26 +VW28	V21.1 = X10 _{sui}
T _{pulse}	Front montant d'activité d'étape	80 bits	VB30..VB39= VW30+VW32+VW34+VW36 +VW38	V31.1=X10↑
T _a	Table d'activation des étape grafcet	80 bits	VB40..VB49= VW40+VW42+VW44+VW46 +VW48+VW46+VW48	V41.1
T _d	Table de désactivation des étapes grafcet	80 bits	VB50..VB59 = VW50+VW52+VW54+VW56 +VW58	V51.1
T _{fl}	Table de forçage à 1 des étapes grafcet	80 bits	VB60..VB69 = VW60+VW62+VW64+VW66 +VW68	V61.1
T _{f0}	Table de forçage à 0 des étapes grafcet	80 bits	VB70..VB79 = VW70+VW72+VW74+VW76 +VW78	V71.1
T _t	Table de transitions	160 bits	VB VB80..VB89= VW80+VW82+VW84+VW86 +VW88 VB90..VB99= VW90+VW92+VW94+VW96 +VW98	t ₁₀ =V81.1 t ₇₄ =V87.4

3.3 Utilisation des tables

La méthode classique dite SET RESET travaillant sur une table d'activité nécessite l'examen global du graphe pour décider de la désactivation et ou l'activation d'une étape. La maintenance d'une machine programmée par cette méthode est délicate.

Dans la méthode proposée nous travaillerons à l'aide de 4 tables. T_p T_s T_a et T_d. Les transitions seront examinées transition par transition et si celle ci est franchissable, nous activons le (les) bit(s) de la table d'activation T_a des étapes postérieures et activons le (les) bit(s) de la table de désactivation des étapes antérieures. Ces activations sont indépendante des autres transitions ce qui simplifie l'écriture et la mise au point.

La priorité à l'activation est alors respectée par l'équation booléenne $T_s = T_p \cdot \bar{T}_d + T_a$

3.4 Adresses particulières

Pour les calculs sur octets(M), mots(W) double mots(D), il est nécessaire de posséder un espace mémoire qui joue le rôle d'accumulateur (au sens microprocesseur).

Adresse	équivalent de MSB a LSB	Taille
VB02	VB02	8 bits
VW02	VB02 , VB03	16 bits
VD02	VB02 , VB03 VB04 , VB05	32 bits



La plage VB06 a VB09 (V6.0 a V9.7) reste libre pour utilisation en bits intermédiaires de calcul (fronts ...)

<i>Adresse</i>	<i>Taille</i>
VB06 .. VB09	32 bits

Soit au total la plage VB00 a VB99 utilisé (100 octets). La CPU en possédant au minimum 1024 (4096 pour la CPU 214) il reste de la place pour tous les traitements annexes!!

dans notre exemple nous prendrons V6.0 pour bit de stockage de l'état antérieur de dcy et V6.1 le bit correspondant à dcy front montant.

L'autre plages mémoire utilisateur plus restreintes (les mémentos M M0.0 à M63.7) par esprit de simplification n'a pas été utilisé dans cette méthode.

IV Architecture logicielle

4.1 possibilités

La CPU 214 ne possède pas toute la richesse de structure des CPU 95 et de celles de la nouvelle série 7 CPU 300. Elles possèdent une structure de sous-programmes (les SBR n) 16 sous-programmes pour la CPU 212 et 64 pour la CPU 214.

Les niveaux d'imbrications de sous programme sont au nombre de 8 maximum

Un sous programme ne peut se rappeler lui-même.

4.2 Architecture proposée

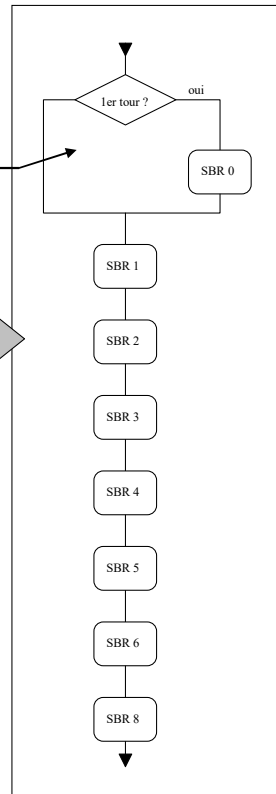
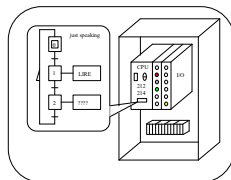


Architecture des unités logicielles

Structure principale

Traitement du
 premier tour
 automate

OB1
 Bloc d'organisation



Le bloc d'organisation cyclique OB1

```

NETWORK1 // Execution du sous
programme SR0 au premier tour automate
0 LD SM0.1
2 CALL K0
NETWORK2 // appel des reactualisations
des tables
5 LD SM0.0
7 CALL K1
NETWORK3 // Appel des calculs
preliminaires
10 LD SM0.0
12 CALL K2
NETWORK4 // Appel des calculs des
transitions
15 LD SM0.0
17 CALL K3
NETWORK5 // Appel des calculs
d'evolution
20 LD SM0.0
22 CALL K4
NETWORK6 // Traitement des forçages
25 LD SM0.0
27 CALL K5
NETWORK7 // Traitements des tempo et
compteurs
30 LD SM0.0
32 CALL K6
NETWORK8 // Combinatoire de sortie
35 LD SM0.0
37 CALL K8
NETWORK9
40 MEND
    
```



<i>Nom</i>	<i>Rôle</i>
SBR 0	Initialisation des graphes à 0 defigage des graphes Création des étapes initiales Initialisation des compteurs tempo, mots en tout genre
SBR 1	Calcul de T_{pulse} Réactualisation des tables d'étapes $T_p=T_s$ Mise à 0 des tables de forçage
SBR 2	Calculs préliminaires (Fronts .. dépend de l'application)
SBR 3	Calcul des réceptivités associées aux transitions (T_i)
SBR 4	Calcul de l'évolution normale des graphes via les tables T_a et T_d Calcul de T_s en fonction de T_a et T_d
SBR 5	Calcul de T_{pulse} Traitement des forçages Calcul de T_s en fonction de T_{fi} et T_{f0} Calcul de T_{pulse}
SBR 6	Traitement des temporisateurs et compteurs
SBR 8	Combinatoire de sortie.
SBR 10	Calcul du front montant des étapes (dans SBR 1, SBR5)

Nota : Les parties en grisés sont indépendantes de l'application.

Le calcul de T_{pulse} doit être fait après chaque modification de la table des états d'activité des étapes (T_s) soit avant le basculement $T_p T_s$ dans le SBR 1, après le calcul d'évolution normale (en début de SBR 5) et après les modifications éventuelles réalisées par les opérations de forçage (en fin de SBR 5).

4.3 Utilisation des bits système

Les bits systèmes suivants sont indispensables

<i>Mnémonique</i>	<i>Rôle</i>
SM0.0	Ressource permanente à 1
SM0.1	Mis à 1 durant le premier tour de calcul

V Détail des unités logicielle

Dans les paragraphes suivants, nous présentons la base des sous programme puis en encadré italique, des exemples correspondant aux graphes exemples de la page 2

5.0 OB1

```

NETWORK1 // Execution du sous programme SR0 au premier tour automate
  0 LD  SM0.1
  2 CALL K0
NETWORK2 // appel des reactualisations des tables
  5 LD  SM0.0
  7 CALL K1
NETWORK3 // Appel des calculs preliminaires
  10 LD  SM0.0
  12 CALL K2
NETWORK4 // Appel des calculs des transitions
    
```

Bit mis a 1 au premier tour automate

Bit toujours à 1



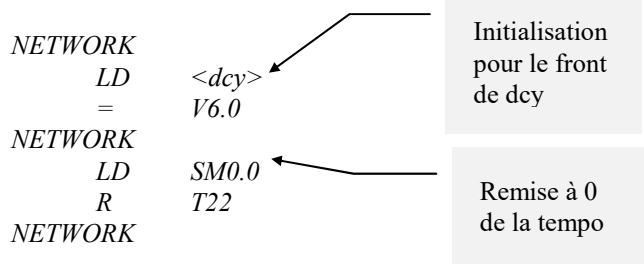
```
15 LD SM0.0
17 CALL K3
NETWORK5 // Appel des calculs d'evolution
20 LD SM0.0
22 CALL K4
NETWORK6 // Traitement des forçages
25 LD SM0.0
27 CALL K5
NETWORK7 // Traitements des tempo et compteurs
30 LD SM0.0
32 CALL K6
NETWORK8 // Combinatoire de sortie
35 LD SM0.0
37 CALL K8
NETWORK9
40 MEND
```

5.1 SBR0

Le rôle de ce sous programme est d'initialiser les graphes et toutes variables système au premier tour automate.

```
NETWORK10 // Sous programme d'initialisation des tables
41 SBR K0
NETWORK11 // Defigage des graphes (mise a 1 de tous les bits
44 LD SM0.0
46 FILL KHFFFF VW0 K1
NETWORK12 // Remplissage de toutes les tables de 0
56 LD SM0.0
58 FILL K0 VW10 K45
NETWORK13 // Mise a 1 des etapes initiales par ex X0
68 LD SM0.0
70 = V10.0
73 = V20.0
NETWORK14 // Initialisation des tempos compteurs ...
76 NOP K0
NETWORK15
78 RET
```

Dans le premier tour automate, il est nécessaire d'initialiser les tempos de mémoriser les états des bits de fronts (dcy)



5.2 SBR 1



```
NETWORK16 // Sous programme de remise a jour des tables
79 SBR K1
NETWORK17 // Calcul de Tpulse
82 LD SM0.0
84 CALL K10
NETWORK18 // Reactualisation de tables d'activite d'etapes Tp = Ts
87 LD SM0.0
89 BMW VW20 VW10 K5
NETWORK19 // Initialisation a 0 des tables Ta Td Tf1 Tf0
99 LD SM0.0
101 FILL K0 VW40 K40
NETWORK20
111 RET
```

Copie de 5 mots
de Ts vers Tp

5.3 SBR 2

```
NETWORK21 // Calculs preliminaires annexes : fronts ...
112 SBR K2
NETWORK22
115 RET
```

Calcul du front de dcy

```
NETWORK
LDN V6.0
U <dcy>
= V6.1
NETWORK
LD <dcy>
= V6.0
NETWORK
```

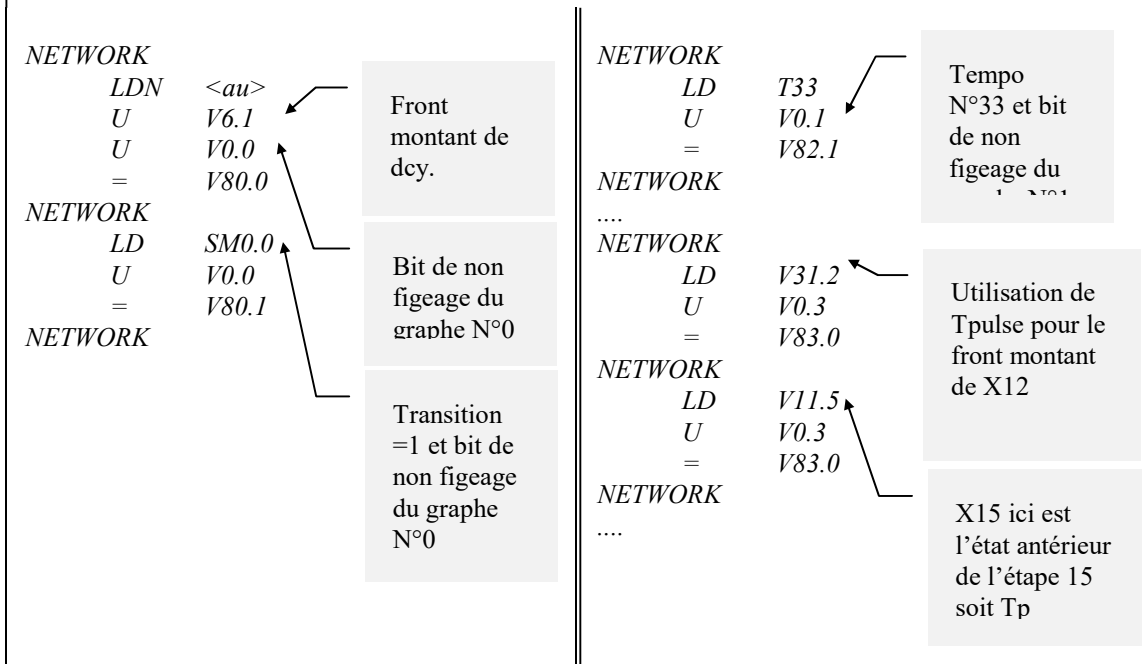
5.4 SBR 3

```
NETWORK23 // Calcul des receptivites des transitions
116 SBR K3
NETWORK24
119 RET
```

Cette unité logicielle est très classique et on peut écrire $T_i = \text{receptivite} \bullet \text{Figeage}$ ou Figeage est le bit de figeage du graphe en question (A 0 si graphe figé, a 1 dans le cas normale). On peut écrire ces équations tant en langage à contact (LADER ou CONT) qu'en langage à instruction (LIST).



Dans le premier tour automate, il est nécessaire d'initialiser les tempos de mémoriser les états des bits de fronts (dcy)



5.5 SBR 4

NETWORK25 // Calculs de l'evolution du graphe

120 SBR K4

NETWORK26 // Traitement transition par transition

123 NOP K0

NETWORK27 // Ts=(Tp et (non Td)) ou Ta

125 LD SM0.0

127 MOVD VD50 VD2

133 INVD VD2

137 UNDD VD10 VD2

143 ORD VD40 VD2

149 MOVD VD2 VD20

155 MOVD VD54 VD2

161 INVD VD2

165 UNDD VD14 VD2

171 ORD VD44 VD2

177 MOVD VD2 VD24

183 MOVW VW58 VW2

189 INW VW2

193 UNDW VW18 VW2

199 ORW VW48 VW2

205 MOVW VW2 VW28

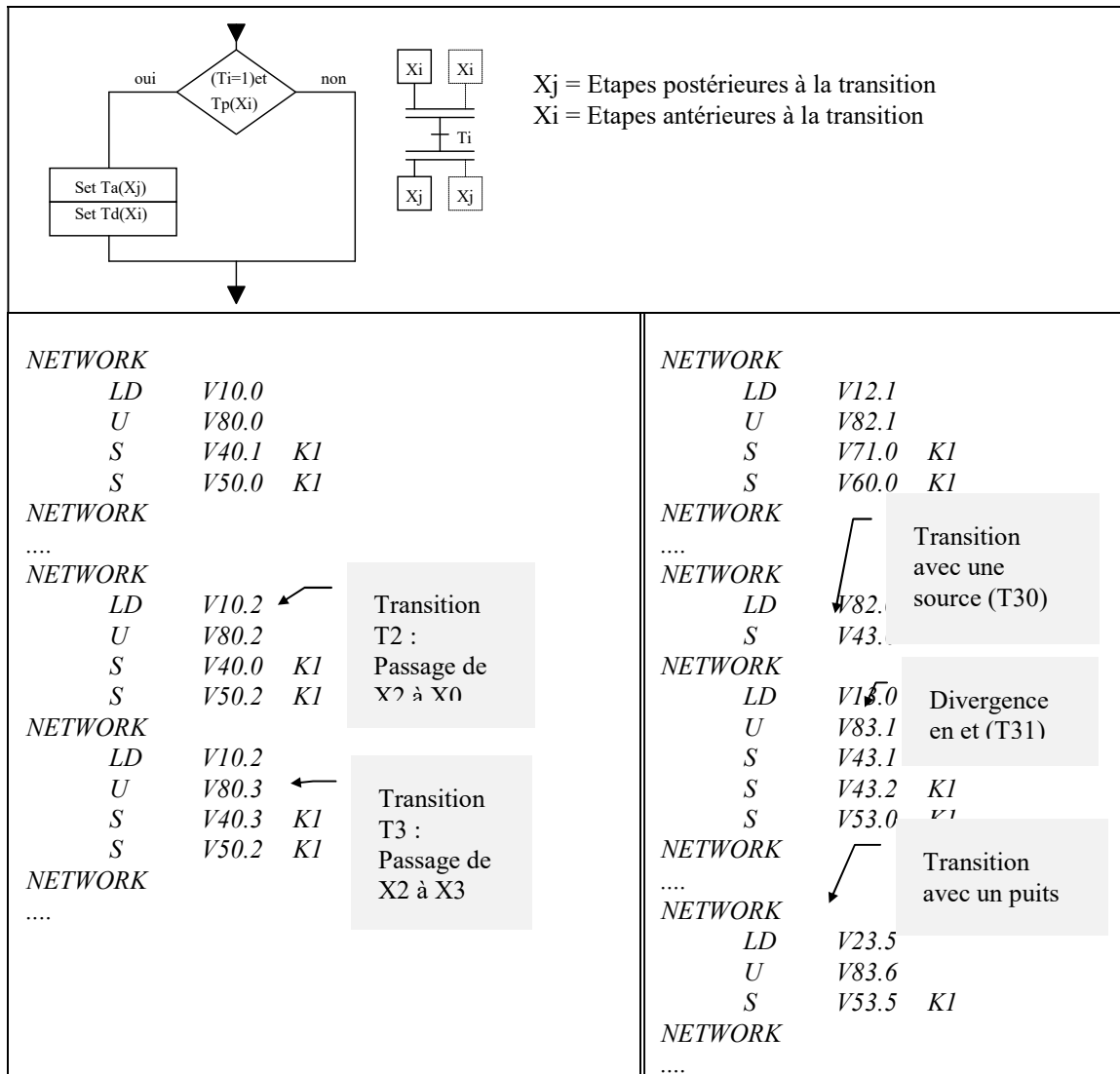
NETWORK28

211 RET

Integrer ici les calculs dépendant de l'application

Il est nécessaire de traiter par paquets de 32 bits puis 16 bits pour effectuer le travail sur les 80 bits des tables

Le traitement transition par transition se résume a la structure suivante



5.6 SBR 5

```

NETWORK30
  212 SBR   K5
NETWORK30 // Calcul de Tpulse
  215 LD    SM0.0
  217 CALL  K10
NETWORK31 // Traitement des forçages étape par étape
  220 NOP   K0
NETWORK32 // Ts=(Ts et (non Tf0)) ou Tf1
  222 LD    SM0.0
  224 MOVD  VD70    VD2
  230 INVD  VD2
  234 UNDD  VD20    VD2
  240 ORD   VD60    VD2
  246 MOVD  VD2     VD20
  252 MOVD  VD74    VD2
  258 INVD  VD2
  262 UNDD  VD24    VD2
  268 ORD   VD64    VD2
    
```

On traite les cas de forçage étape par étape en activant Tf0 et Tf1

Ici aussi on traite par paquets pour arriver aux 80



```

274 MOVD VD2 VD24
280 MOVW VW78 VW2
286 INWV VW2
290 UNDW VW28 VW2
296 ORW VW68 VW2
302 MOVW VW2 VW28
NETWORK33 // Calcul de Tpulse
308 LD SM0.0
310 CALL K10
NETWORK34
313 RET
    
```

On traite les forçages étape par étape. Comme il est possible de forcer plusieurs bits d'un coup, il est intéressant de 'caler' nos graphes sur des frontières de byte au niveau des tables.

```

NETWORK
LD V20.0
S V71.0 K8
S V72.0 K2
NETWORK
LD V20.1
S V71.0 K8
S V72.0 K2
S V61.0 K1
NETWORK
..
    
```

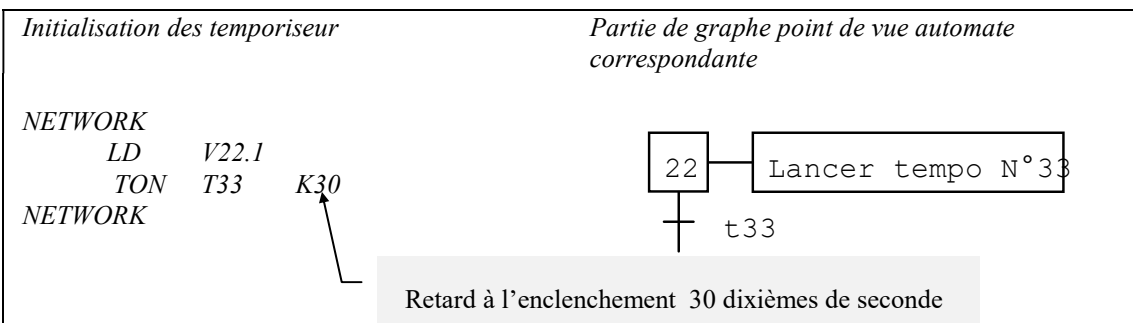
Dans l'étape 0, on a forçage à 0 de toutes les étapes du graphe N°1

Dans l'étape 1 on a forçage à 0 de toutes les étapes du graphe N°1 et forçage à 1 de l'étape N°10

5.7 SBR 6

```

NETWORK36 // Traitement des temporisateurs et compteurs
314 SBR K6
NETWORK37
317 RET
    
```



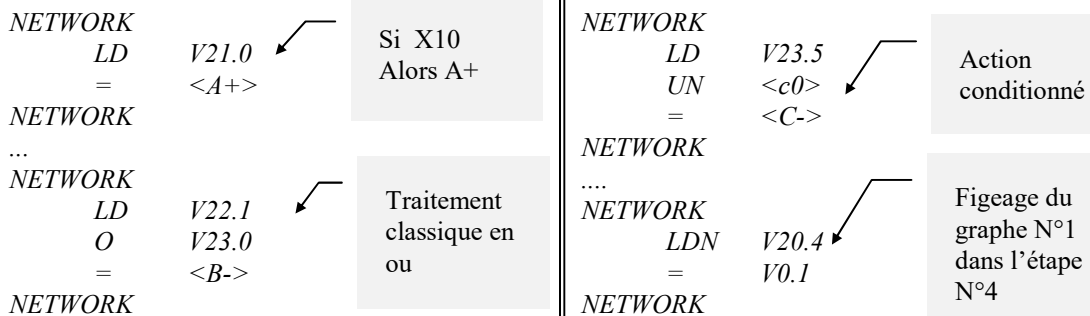
5.8 SBR 8

```

NETWORK38 // Combinatoire de sortie
318 SBR K8
NETWORK39
321 RET
    
```



On traite classiquement en étudiant de manière globale les sorties. Les figeages sont des sorties particulières



5.9 SBR 10

Ce sous programme évite de répéter des lignes fastidieuses. Il calcul la table T_{pulse} en fonction de T_p et T_s .

```

NETWORK39 // Debut du sous programme SR10 de calcul de Tpulse
322 SBR K10
NETWORK40
325 LD SM0.0
327 MOVD VD10 VD2
333 INVD VD2
337 UNDD VD20 VD2
343 MOVD VD2 VD30
349 MOVD VD14 VD2
355 INVD VD2
359 UNDD VD24 VD2
365 MOVD VD2 VD34
371 MOVW VW18 VW2
377 INVW VW2
381 UNDW VW28 VW2
387 MOVW VW2 VW38
NETWORK41
393 RET
    
```

$T_{pulse} = T_s$ et (non T_p)

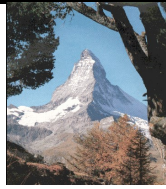
VI Taille mémoire

La taille minimum de cette structure (méthode) est de ~390 octets. La programmation de diverses applications montre que en moyenne l'ajout d'une étape conduit à une augmentation de la taille programme de 45 octets.

La CPU 214 possède 4096 octets de mémoire programme on peut donc espérer implémenter des applications de $(4096-390) \text{ Div } 45 = 82$ étapes sur une CPU 214, et, $(1024-387) \text{ div } 45 = 14$ étapes pour une CPU 212. La structure de base étant prévue pour 80 étapes je vous laisse le soins de la conclusion quand au choix de votre CPU.

Exemple d'implémentation du GRAFCET sur automate SIEMENS CPU 212 et 214

Exemple N°1

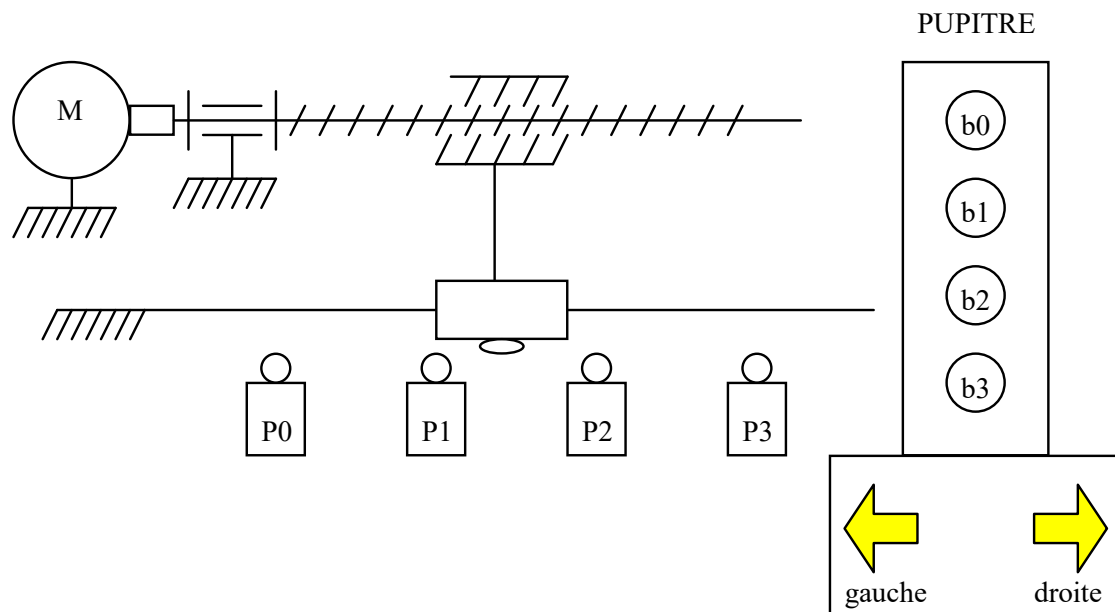


Cordier
Ridacker
Le 30/09/2024

1/9

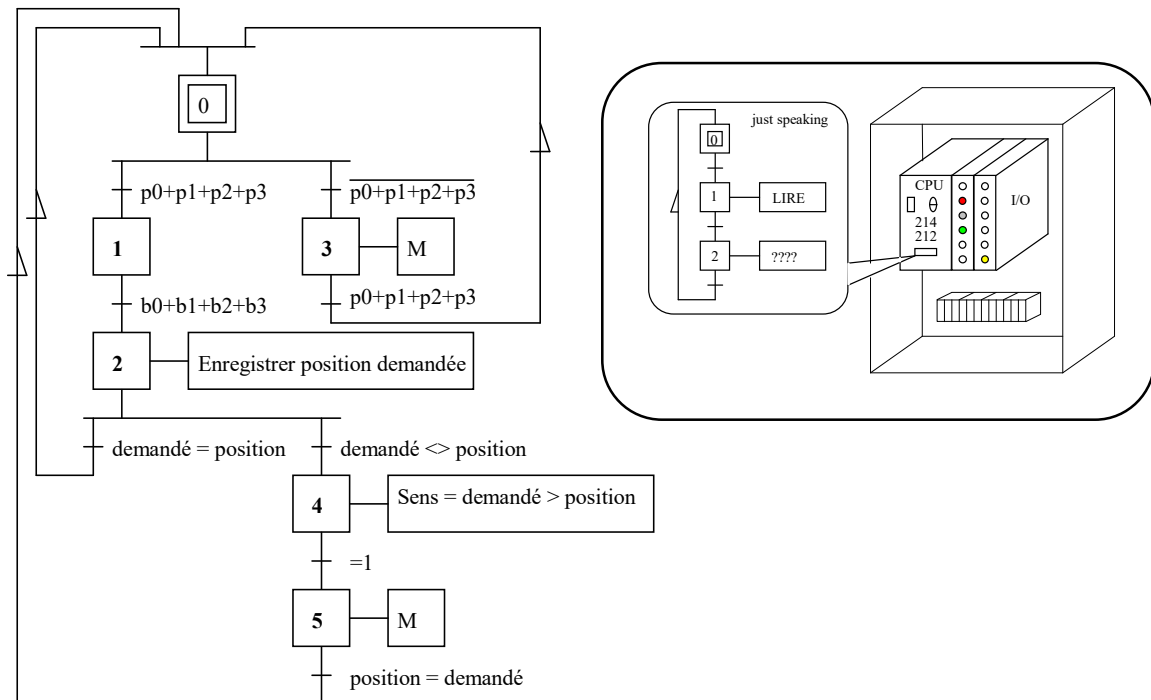
L'exemple ci-dessous a été développé pour mettre en oeuvre les concepts développés dans le cas d'un automatisme très simple limité à un graphe.

Nous allons étudier le fonctionnement d'un axe de positionnement piloté par un moteur à double sens de marche tel que décrit ci-dessous.





Graphe Point de vue système



GRAFCET du point de vue P.C.

Pour automate de type list et sans outils graphiques

Graphes	Sorties, voyant	Entrées	pupitre
G0 = GPN	A0.0 = Marche Moteur	E0.0 = Pos N°0	E04 = demande position 0
	A0.1 = Sens Moteur	E0.1 = Pos N°1	E05 = demande position 1
	A0.2 = Voyant Droite	E0.2 = Pos N°2	E0.6 = demande position 2
	A0.3 = Voyant Gauche	E0.3 = Pos N°3	E0.7 = demande position 3

Affectation des mots internes

V6.0 : Sens de déplacement

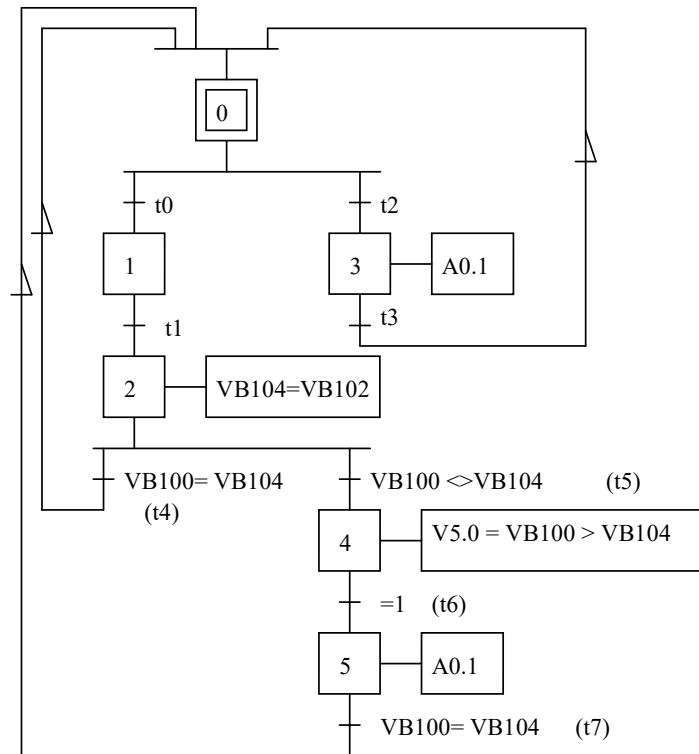
L'unité logicielle SBR **SBR 11** calcul les deux mots **VB100** et **VB102**

V00.0	V100.1	V100.2	V100.3
=1 si p ₀ actionné	=1 si p ₁ actionné	=1 si p ₂ actionné	=1 si p ₃ actionné

V102.0	V102.1	V102.2	V102.3
=1 si b ₀ actionné	=1 si b ₁ actionné	=1 si b ₂ actionné	=1 si b ₃ actionné



Dans l'étape N°2 on sauve la valeur VB102 dans **VB104** pour utilisation ultérieure dans les transitions.



$$t0 = E0.0 + E0.1 + E0.2 + E0.3$$

$$t1 = E0.4 + E0.5 + E0.6 + E0.7$$

$$t2 = /t4$$

$$t3 = t4$$

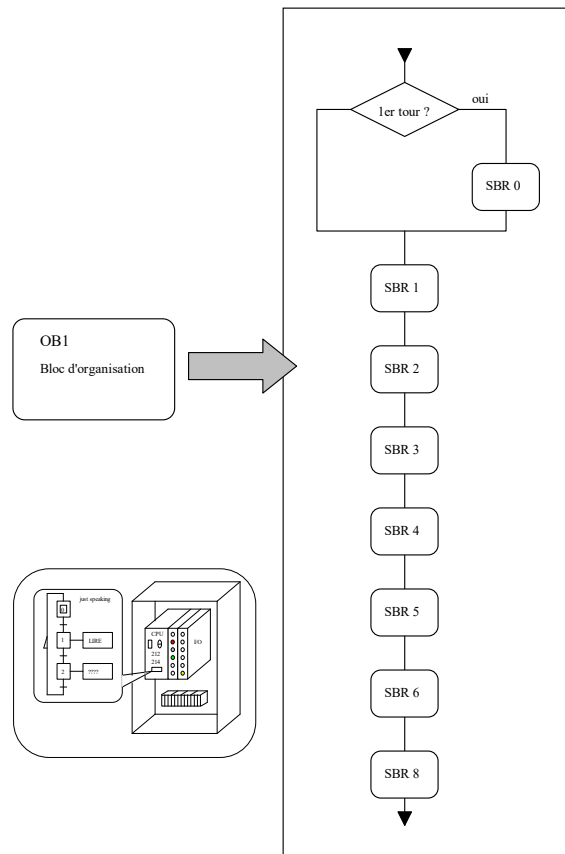
$$A0.2 = (X3 + X5) \cdot V5.0$$

$$A0.3 = (X3 + X5) / V5.0$$

Le programme



```
NETWORK1 // Execution du sous programme
SR0 au premier tour automate
  0 LD SM0.1
  2 CALL K0
NETWORK2 // appel des reactualisations des
tables
  5 LD SM0.0
  7 CALL K1
NETWORK3 // Appel des calculs
preliminaires
  10 LD SM0.0
  12 CALL K2
NETWORK4 // Appel des calculs des
transitions
  15 LD SM0.0
  17 CALL K3
NETWORK5 // Appel des calculs d'evolution
  20 LD SM0.0
  22 CALL K4
NETWORK6 // Traitement des forcages
  25 LD SM0.0
  27 CALL K5
NETWORK7 // Traitements des tempo et
compteurs
  30 LD SM0.0
  32 CALL K6
NETWORK8 // Combinatoire de sortie
  35 LD SM0.0
  37 CALL K8
NETWORK9
  40 MEND
```



Le sous-programme SBR 0

```
NETWORK10 // Sous programme d'initialisation des
tables
  41 SBR K0
NETWORK11 // Remplissage de toutes les tables de 0
  44 LD SM0.0
  46 FILL K0 VW10 K45
NETWORK12 // Mise a 1 des etapes initiales par ex X0
  56 LD SM0.0
  58 = V10.0
  61 = V20.0
NETWORK13 // Initialisation des tempos compteurs ...
  64 LD SM0.0
  66 MOVW K0 VW104
  72 MOVW K0 VW102
  78 MOVW K0 VW100
NETWORK14
  84 RET
```



Le sous-programme SBR 1

```
NETWORK15 // Sous programme de remise a jour des
tables
  85 SBR K1
NETWORK16 // Calcul de Tpulse
  88 CALL K10
NETWORK17 // Reactualisation de tables d'activite
d'etapes Tp = Ts
  91 LD SM0.0
  93 BMW VW20 VW10 K5
NETWORK18 // Initialisation a 0 des tables Ta Td Tf1
Tf0
  103 LD SM0.0
  105 FILL K0 VW40 K40
NETWORK19
  115 RET
```

Le sous-programme SBR 2

```
NETWORK20 // Calculs preliminaires annexes : fronts .
  116 SBR K2
NETWORK21
  119 CALL K11
NETWORK22
  122 RET
```

Appel traitement des
divers capteurs

Le sous-programme SBR 3

```
NETWORK23 // Calcul des receptivites des transitions
  123 SBR K3
NETWORK24
  126 LD V0.0
  129 UB>= VB100 K1
  134 = V80.0
NETWORK25
  137 LD V0.0
  140 U V80.0
  143 = V80.2
NETWORK26
  146 LD V0.0
  149 UB>= VB102 K1
  154 = V80.1
NETWORK27
  157 LD V0.0
  160 U V80.0
  163 = V80.3
```

```
NETWORK28
  166 LD V0.0
  169 UB= VB100 VB104
  175 = V80.4
NETWORK29
  178 LD V0.0
  181 UN V80.4
  184 = V80.5
NETWORK30
  187 LD V0.0
  190 U SM0.0
  192 = V80.6
NETWORK31
  195 LD V0.0
  198 U V80.4
  201 = V80.7
NETWORK32
  204 RET
```



Le sous-programme SBR 4

NETWORK33 // Calculs de l'évolution du graphe

205 SBR K4

NETWORK34 // Traitement transition par transition

208 LD V80.0

211 U V20.0

214 S V40.1 KI

221 S V50.0 KI

NETWORK35

228 LD V80.1

231 U V20.1

234 S V40.2 KI

241 S V50.1 KI

NETWORK36

248 LD V80.2

251 U V20.0

254 S V40.3 KI

261 S V50.0 KI

NETWORK37

268 LD V80.3

271 U V20.3

274 S V40.0 KI

281 S V50.3 KI

NETWORK38

288 LD V80.4

291 U V20.2

294 S V40.0 KI

301 S V50.2 KI

NETWORK39

308 LD V80.5

311 U V20.2

314 S V40.4 KI

321 S V50.2 KI

NETWORK40

328 LD V80.6

331 U V20.4

334 S V40.5 KI

341 S V50.4 KI

NETWORK41

348 LD V80.7

351 U V20.5

354 S V40.0 KI

361 S V50.5 KI

NETWORK42 // $T_s = (T_p \text{ et (non } T_d)) \text{ ou } T_a$

368 LD SM0.0

370 MOVD VD50 VD2

376 INVD VD2

380 UNDD VD10 VD2

386 ORD VD40 VD2

392 MOVD VD2 VD20

398 MOVD VD54 VD2

404 INVD VD2

408 UNDD VD14 VD2

414 ORD VD44 VD2

420 MOVD VD2 VD24

426 MOVW VW58 VW2

432 INVW VW2

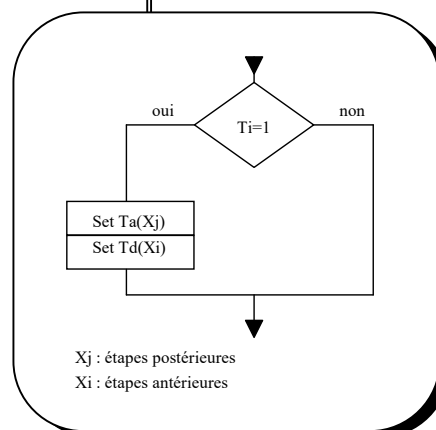
436 UNDW VW18 VW2

442 ORW VW48 VW2

448 MOVW VW2 VW28

NETWORK43

454 RET



Le sous-programme SBR 5

NETWORK44 // Traitement des forçages

455 SBR K5

NETWORK45 // Calcul de Tpulse

458 LD SM0.0

460 CALL K10

NETWORK46 // Traitement des forçages etape par etape

501 INVD VD2

505 UNDD VD24 VD2

511 ORD VD64 VD2

517 MOVD VD2 VD24

523 MOVW VW78 VW2

529 INVW VW2



```
463 NOP K0
NETWORK47 // Ts=(Ts et (non Tf0)) ou Tf1
465 LD SM0.0
467 MOVD VD70 VD2
473 INVD VD2
477 UNDD VD20 VD2
483 ORD VD60 VD2
489 MOVD VD2 VD20
495 MOVD VD74 VD2
```

```
533 UNDW VW28 VW2
539 ORW VW68 VW2
545 MOVW VW2 VW28
NETWORK48 // Calcul de Tpulse
551 LD SM0.0
553 CALL K10
NETWORK49
556 RET
```

Le sous-programme SBR 6

```
NETWORK50 // Tempos compteurs
557 SBR K6
NETWORK51
560 RET
```

Le sous-programme SBR 8

```
NETWORK52 // Combinatoire de sortie
561 SBR K8
NETWORK53
564 LD V20.3
567 O V20.5
570 = A0.0
NETWORK54
572 LD V20.4
575 UB>= VB102 VB100
581 = V6.0
584 = A0.1
NETWORK55
586 LD V20.3
589 O V20.5
592 U V6.0
595 = A0.2
```

```
NETWORK56
597 LD V20.3
600 O V20.5
603 UN V6.0
606 = A0.3
NETWORK57
608 LD V20.2
611 MOVB VB102 VB104
NETWORK58
617 RET
```

Le sous-programme SBR 10

```
NETWORK59 // Debut du sous programme SR10 de calcul de Tpulse
618 SBR K10
NETWORK60
621 LD SM0.0
623 MOVD VD10 VD2
629 INVD VD2
633 UNDD VD20 VD2
639 MOVD VD2 VD30
645 MOVD VD14 VD2
651 INVD VD2
```

Tpulse = Ts et non Tp

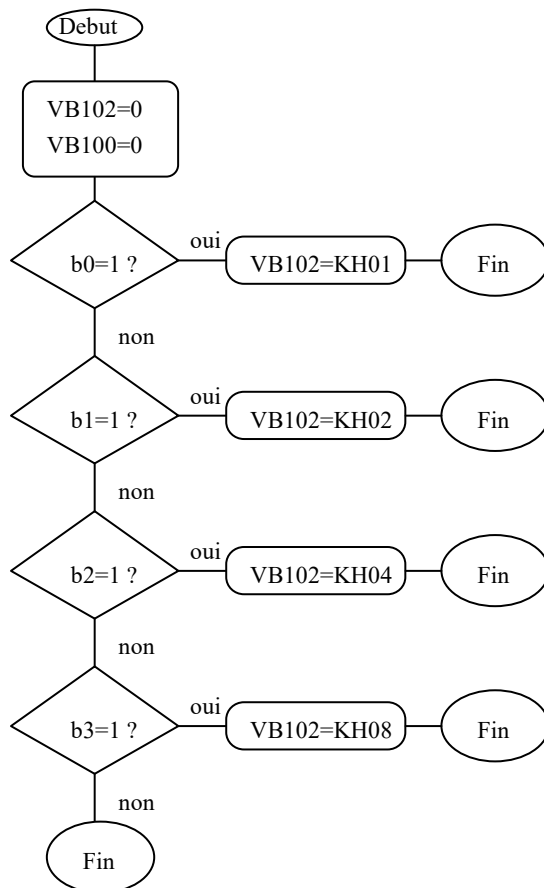


```

655 UNDD VD24 VD2
661 MOVD VD2 VD34
667 MOVW VW18 VW2
673 INVW VW2
677 MOVW VW2 VW38
NETWORK61
683 RET

```

Le sous-programme SBR 11



```

NETWORK62 // Traitement special des entrees
E0.0 a E0.7
684 SBR K11
NETWORK63
687 LD SM0.0
689 R V100.0 K7
696 R V102.0 K7
NETWORK64
703 LD E0.4
705 S V100.0 K1
712 CRET
NETWORK65

```

```

NETWORK66
723 LD E0.6
725 S V100.2 K1
732 CRET
NETWORK67
733 LD E0.7
735 S V100.3 K1
742 CRET
NETWORK68
743 LD SM0.0
745 MOVB EB0 VB2
751 UNDW KHF00 VW2

```

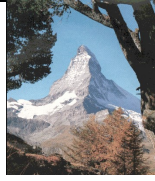


713 LD E0.5
715 S V100.1 K1
722 CRET

757 MOVB VB2 VB102
NETWORK69
763 RET

Exemple d'implementation du GRAFCET sur automate SIEMENS CPU 212 et 214

Exemple N°2



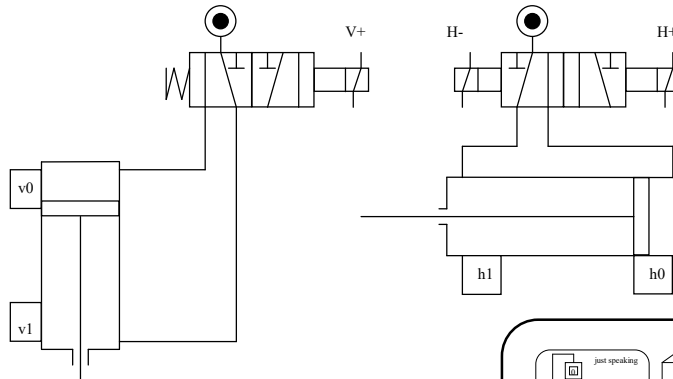
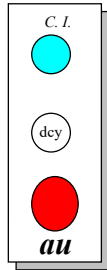
Cordier

Ridacker

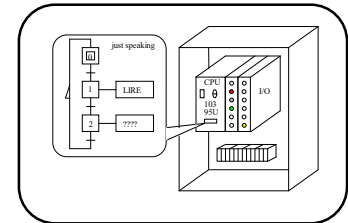
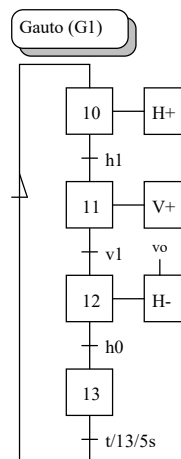
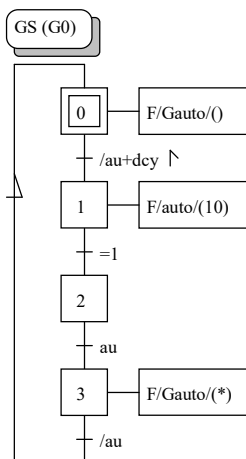
Le 30/09/2024

1/7

Manipulateur en L

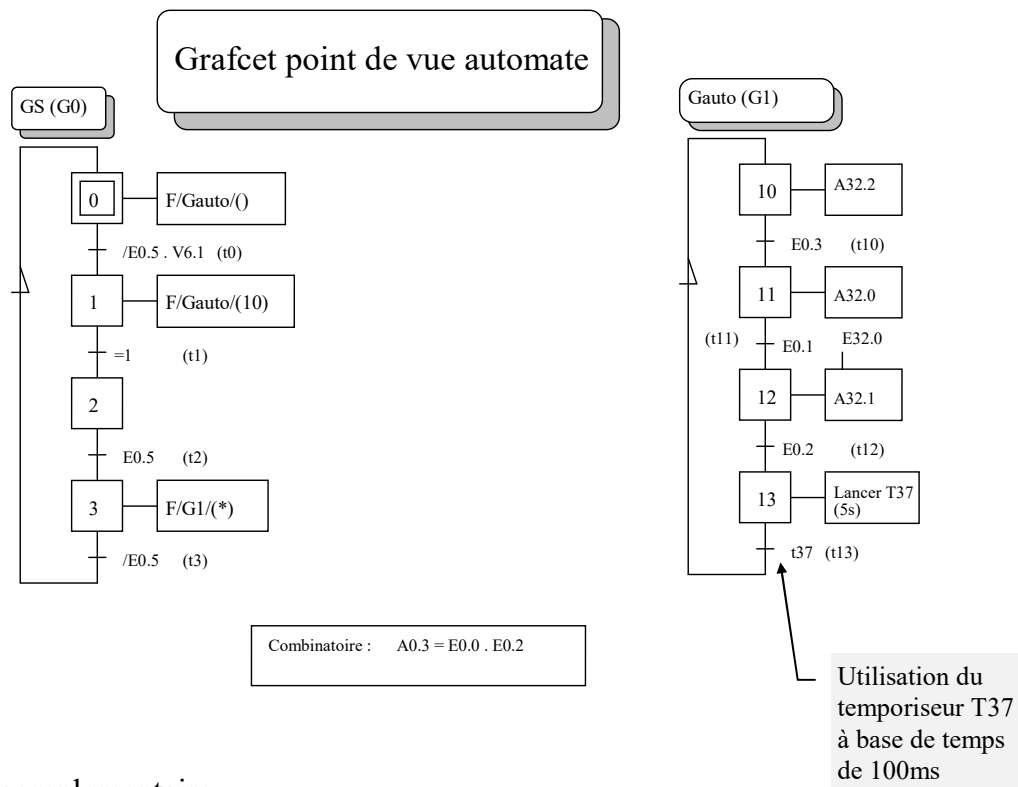


Les conditions initiales sont v0 et h0



Affectation de Entrées - Sorties

Graphes	Sorties	Entrées	pupitre
G0 = GS	A0.0 = V+	E0.0 = v0	E0.4 = dcy
	A0.1 = H-	E0.1 = v1	E0.5 = au
G1 = Gauto	A0.2 = H+	E0.2 = h0	
	A0.3 = Voyant CI	E0.3 = h1	



Adressage complémentaire

Pour simplifier la programmation et la lecture, nous avons adopté une numérotation décimale qui se rapproche au plus près de la numération hexadécimale. Les adresses et numéros 8 a 9 ne sont pas utilisés.

Au niveau des tables, nous obtenons alors l'équivalence des bits dans la zone de données V_n ($n \in [0..1023]$ pour CPU 212 ; $n \in [0..4095]$ pour CPU 214)

pour les numéro d'étapes :

V	0	1	2	10	11	12	13
T_p	10.0	10.1	10.2	11.0	11.1	11.2	11.3
T_s	20.0			21.0			
T_{pulse}	30.0			31.0			
T_a	40.0			41.0			
T_d	50.0			51.0			
T_{fl}	60.0			61.0			
T_{f0}	70.0			71.0			

pour les transitions :

V	0	1	2	3	10	11	12	13
T_t	80.0	80.1	80.2	80.3	81.0	81.1	81.2	81.3



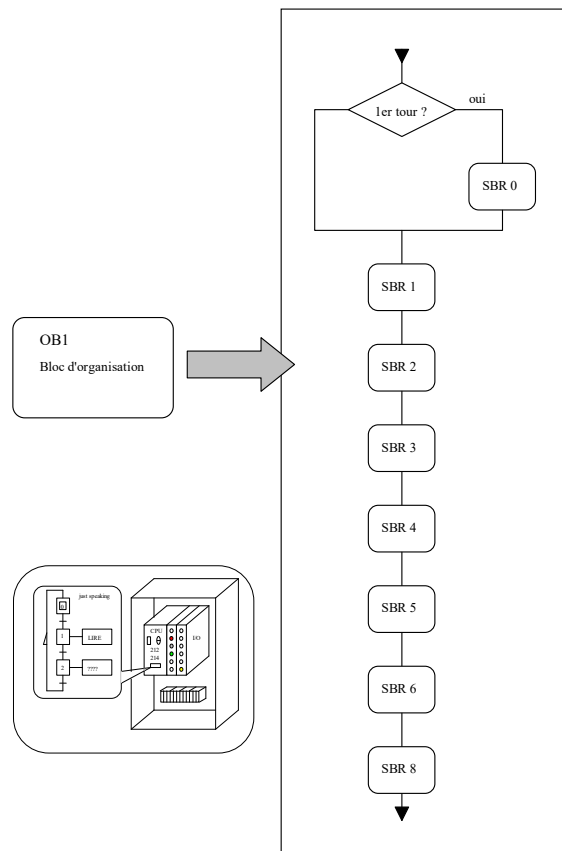
Le bit **V6.0** est utilisé pour le stockage de l'état de dcy.
Le bit **V6.1** correspond au front montant de dcy.

Programme

Dans le listing qui suit les parties en grisées sont indépendantes de l'application.

Programme principal OB1

```
NETWORK1 // Execution du sous programme SR0  
au premier tour automate  
0 LD SM0.1  
2 CALL K0  
NETWORK2 // appel des reactualisations des tables  
5 LD SM0.0  
7 CALL K1  
NETWORK3 // Appel des calculs  
preliminaires  
10 LD SM0.0  
12 CALL K2  
NETWORK4 // Appel des calculs des  
transitions  
15 LD SM0.0  
17 CALL K3  
NETWORK5 // Appel des calculs d'evolution  
20 LD SM0.0  
22 CALL K4  
NETWORK6 // Traitement des forcages  
25 LD SM0.0  
27 CALL K5  
NETWORK7 // Traitements des tempo et  
compteurs  
30 LD SM0.0  
32 CALL K6  
NETWORK8 // Combinatoire de sortie  
35 LD SM0.0  
37 CALL K8  
NETWORK9  
40 MEND
```



Sous Programme SBR 0

```
NETWORK10 // Sous programme d'initialisation des tables  
41 SBR K0  
NETWORK11 // Defigage des graphes (mise a 1 de tous les bits  
44 LD SM0.0  
46 FILL KHFFFF VW0 K1  
NETWORK12 // Remplissage de toutes les tables de 0
```



```
56 LD SM0.0
58 FILL K0 VW10 K45
NETWORK13 // Mise a 1 des etapes initiales par ex X0
68 LD SM0.0
70 = V10.0
73 = V20.0
NETWORK14 // Mise en memoire de dcy
76 LD E0.4
78 = V6.0
NETWORK15 // Initialisation des tempos compteurs ...
81 LD SM0.0
83 R T110 K1
NETWORK16
90 RET
```

Sous Programme SBR 1

```
NETWORK17 // Sous programme de remise a jour des tables
91 SBR K1
NETWORK18 // Calcul de Tpulse
94 CALL K10
NETWORK19 // Reactualisation de tables d'activite d'etapes Tp = Ts
97 LD SM0.0
99 BMW VW20 VW10 K5
NETWORK20 // Initialisation a 0 des tables Ta Td Tf1 Tf0
109 LD SM0.0
111 FILL K0 VW40 K40
NETWORK21
121 RET
```

Sous Programme SBR 2

```
NETWORK22 // Calculs preliminaires annexes : fronts ...
122 SBR K2
NETWORK23 // Front montant de dcy
125 LD E0.4
127 UN V6.0
130 = V6.1
NETWORK24
133 LD E0.4
135 = V6.0
NETWORK25
138 RET
```

Sous Programme SBR 3

```
NETWORK26 // Calcul des receptivites des transitions NETWORK31
139 SBR K3 177 LD E0.3
NETWORK27 179 U V0.1
```



```
142 LDN E0.5
144 U V6.1
147 U V0.0
150 = V80.0
NETWORK28
153 LD SM0.0
155 U V0.0
158 = V80.1
NETWORK29
161 LD E0.5
163 U V0.0
166 = V80.2
NETWORK30
169 LDN E0.5
171 U V0.0
174 = V80.3
```

```
182 = V81.0
NETWORK32
185 LD E0.1
187 U V0.1
190 = V81.1
NETWORK33
193 LD E0.2
195 U V0.1
198 = V81.2
NETWORK34
201 LD T37
203 U V0.1
206 = V81.3
NETWORK35
209 RET
```

Sous Programme SBR 4

NETWORK36 // Calculs de l'évolution du graphe

210 SBR K4

NETWORK37 // Traitement transition par transition

213 LD V10.0

216 U V80.0

219 S V40.1 KI

226 S V50.0 KI

NETWORK38

233 LD V10.1

236 U V80.1

239 S V40.2 KI

246 S V50.1 KI

NETWORK39

253 LD V10.2

256 U V80.2

259 S V40.3 KI

266 S V50.2 KI

NETWORK40

273 LD V10.3

276 U V80.3

279 S V40.0 KI

286 S V50.3 KI

NETWORK41

293 LD V11.0

296 U V81.0

299 S V41.1 KI

306 S V51.0 KI

NETWORK42

313 LD V11.1

316 U V81.1

319 S V41.2 KI

326 S V51.1 KI

NETWORK43

NETWORK44

353 LD V11.3

356 U V81.3

359 S V41.0 KI

366 S V51.3 KI

NETWORK45 // $T_s = (T_p \text{ et (non } T_d)) \text{ ou } T_a$

373 LD SM0.0

375 MOVD VD50 VD2

381 INVD VD2

385 UNDD VD10 VD2

391 ORD VD40 VD2

397 MOVD VD2 VD20

403 MOVD VD54 VD2

409 INVD VD2

413 UNDD VD14 VD2

419 ORD VD44 VD2

425 MOVD VD2 VD24

431 MOVW VW58 VW2

437 INVW VW2

441 UNDW VW18 VW2

447 ORW VW48 VW2

453 MOVW VW2 VW28

NETWORK46

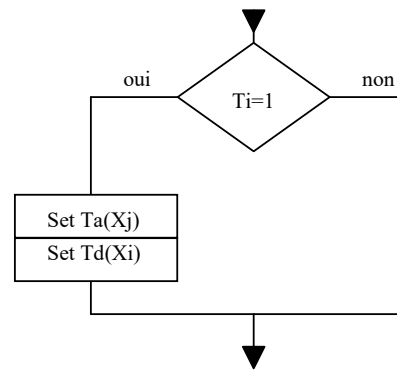
459 RET



```

333 LD    V11.2
336 U     V81.2
339 S     V41.3    K1
346 S     V51.2    K1

```



Xj : étapes postérieures

Xi : étapes antérieures

Sous Programme SBR 5

```

NETWORK47 // Traitement general des forçages
460 SBR    K5

```

```

NETWORK48 // Calcul de Tpulse

```

```

463 LD     SM0.0

```

```

465 CALL   K10

```

```

NETWORK49 // Traitement des forçages etape par etape

```

```

468 LD     V20.0

```

```

471 S      V70.0    K4

```

```

NETWORK50

```

```

478 LD     V20.1

```

```

481 S      V70.0    K4

```

```

488 S      V60.0    K1

```

```

NETWORK51 // Ts=(Ts et (non Tf0)) ou Tf1

```

```

495 LD     SM0.0

```

```

497 MOVD   VD70     VD2

```

```

503 INVD   VD2

```

```

507 UNDD   VD20     VD2

```

```

513 ORD    VD60     VD2

```

```

519 MOVD   VD2      VD20

```

```

525 MOVD   VD74     VD2

```

```

531 INVD   VD2

```

```

535 UNDD   VD24     VD2

```

```

541 ORD    VD64     VD2

```

```

547 MOVD   VD2      VD24

```

```

553 MOVW   VW78     VW2

```

```

559 INVW   VW2

```

```

563 UNDW   VW28     VW2

```

```

569 ORW    VW68     VW2

```

```

575 MOVW   VW2      VW28

```

```

NETWORK52 // Calcul de Tpulse

```

```

581 LD     SM0.0

```

```

583 CALL   K10

```

```

NETWORK53

```

```

586 RET

```

Sous Programme SBR 6

```

NETWORK54 // Compteurs / Tempos

```

```

587 SBR    K6

```

```

NETWORK55

```

```

590 LD     V21.3

```

```

593 TON    T37      K50

```

```

NETWORK56

```

```

599 RET

```



Sous Programme SBR 8

NETWORK57 // Combinatoire general de sortie

600 SBR K8

NETWORK58

603 LD V21.0

606 = A0.2

NETWORK59

608 LD V21.1

611 = A0.0

NETWORK60

613 LD V21.2

616 U E0.0

618 = A0.1

NETWORK61

620 LDN V20.3

623 = V0.1

NETWORK62

626 RET

Sous Programme SBR 10

NETWORK63 // Debut du sous programme SR10 de calcul de Tpulse

627 SBR K10

NETWORK64

630 LD SM0.0

632 MOVD VD10 VD2

638 INVD VD2

642 UNDD VD20 VD2

648 MOVD VD2 VD30

654 MOVD VD14 VD2

660 INVD VD2

664 UNDD VD24 VD2

670 MOVD VD2 VD34

676 MOVW VW18 VW2

682 INVW VW2

686 MOVW VW2 VW38

NETWORK65

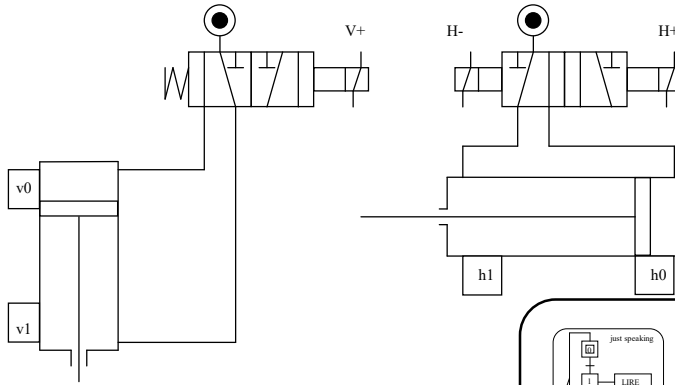
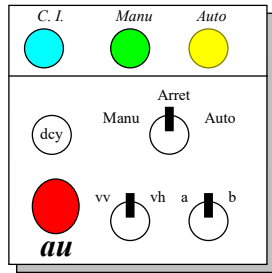
692 RET

Implementation du GRAFCET sur automate SIEMENS CPU 212 et 214 exemple N°3

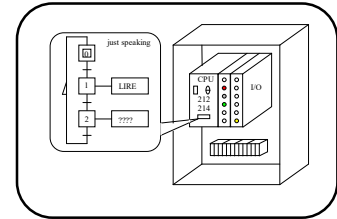


Cordier
Ridacker
Le 30/09/2024

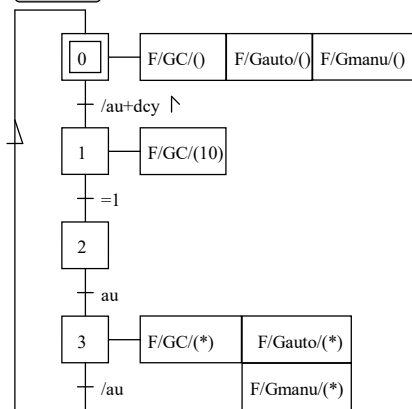
Manipulateur en L



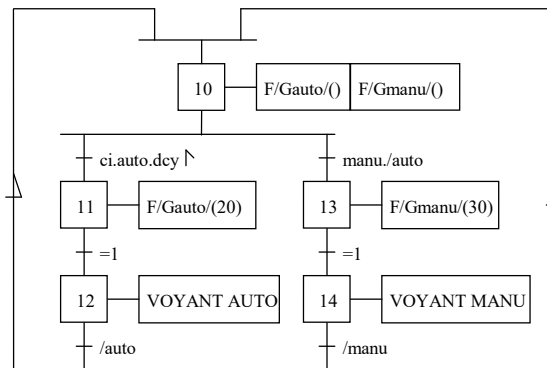
Les conditions initiales sont v0 et h0



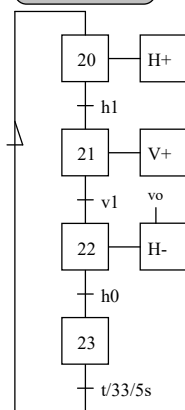
GS (G0)



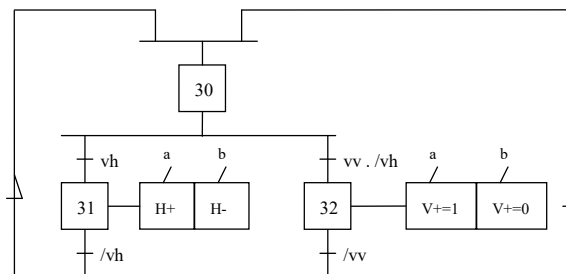
GC (G1)



Gauto (G2)



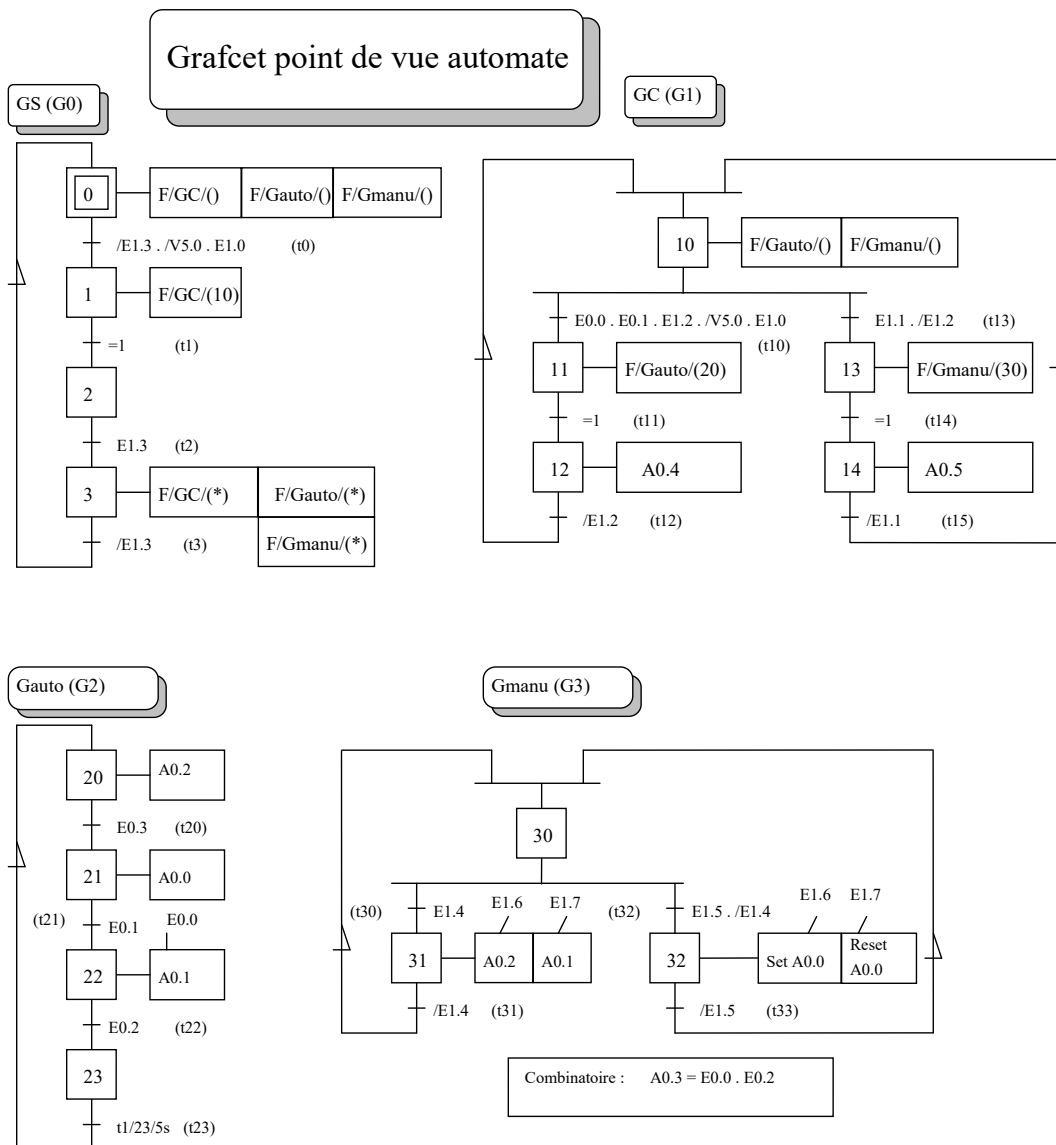
Gmanu (G3)





Affectation de Entrées - Sorties

Graphes	Sorties	Entrées	pupitre	
G0 = GS	A0.0 = V+	E0.0 = v0	E1.0 = dcy	E1.4 = vh
G1 = GC	A0.1 = H-	E0.1 = v1	E1.1 = manu	E1.5 = vv
G2 = Gauto	A0.2 = H+	E0.2 = h0	E1.2 = auto	E1.6 = a
G3 = Gmanu	A0.3 = Voyant CI	E0.3 = h1	E1.3 = au	E1.7 = b
	A0.4 = Voyant Auto			
	A0.5 = Voyant Manu			





Adressage complémentaire

Pour simplifier la programmation et la lecture, nous avons adopté une numérotation décimale qui se rapproche au plus près de la numération hexadécimale. Les adresses et numéros 8 à 9 ne sont pas utilisés.

Au niveau des tables, nous obtenons alors l'équivalence des bits dans la zone de données Vn ($n \in [0..1023]$ pour CPU 212 ; $n \in [0..4095]$ pour CPU 214)

pour les numéro d'étapes :

V	0	1	2	3	10	11	12	13	14	20	21	22	23	30	31	32
T_p	10.0	10.1	10.2	10.3	11.0	11.1	11.2	11.3	11.4	12.0	12.1	12.2	12.3	13.0	13.1	13.2
T_s	20.0				21.0					22.0				23.0		
T_{pulse}	30.0				31.0					32.0				33.0		
T_a	40.0				41.0					42.0				43.0		
T_d	50.0				51.0					52.0				53.0		
T_{fl}	60.0				61.0					62.0				63.0		
T_{f0}	70.0				71.0					72.0				73.0		

pour les transitions :

V	0	1	2	3	10	11	12	13	14	15	20	21	22	23
T_t	80.0	80.1	80.2	80.3	81.0	81.1	81.2	81.3	81.4	81.5	82.0	82.1	82.2	82.3

V	30	31	32	33
T_t	83.0	83.1	83.2	83.3

Le bit $V5.0$ est utilisé pour le calcul du front montant de dcv.

Programme

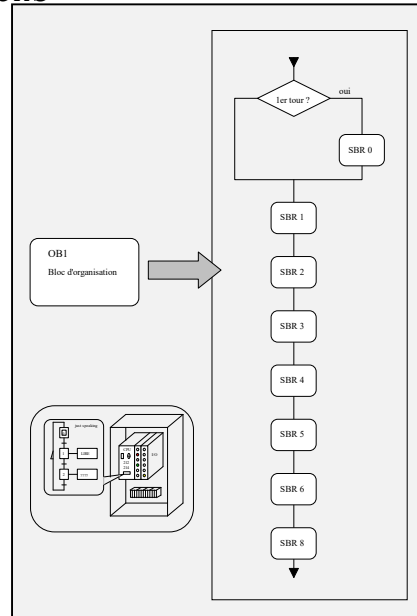
Dans le listing qui suit les parties en italiques sont indépendantes de l'application.

Programme principal

```
NETWORK1 // Execution du sous programme
          SR0 au premier tour
          automate
```



```
0 LD SM0.1
2 CALL K0
NETWORK2 // appel des reactualisations
des tables
5 LD SM0.0
7 CALL K1
NETWORK3 // Appel des calculs
preliminaires
10 LD SM0.0
12 CALL K2
NETWORK4 // Appel des calculs des
transitions
15 LD SM0.0
17 CALL K3
NETWORK5 // Appel des calculs
d'evolution
20 LD SM0.0
22 CALL K4
NETWORK6 // Traitement des forcages
25 LD SM0.0
```





```
27 CALL K5
NETWORK7 // Traitements des tempo et compteurs
30 LD SM0.0
32 CALL K6
NETWORK8 // Combinatoire de sortie
35 LD SM0.0
37 CALL K8
NETWORK9
40 MEND
```

Sous Programme SBR 0

```
NETWORK10 // Sous programme d'initialisation des tables
41 SBR K0
NETWORK11 // Defigeage des graphes (mise a 1 des 16 bits)
44 FILL KHFFFF VW0 K1
NETWORK12 // Remplissage de toutes les tables de 0
54 FILL K0 VW10 K45
NETWORK13 // Mise a 1 de l'etape initiale X0
64 LD SM0.0
66 = V10.0
67 = V20.0
NETWORK14 // utilise pour le front de dcy (E1.0)
69 LD E1.0
71 = V5.0
NETWORK15
74 RET
```

← A l'initialisation, le système est réputé satable depuis longtemps on initialise donc T_p et T_s

Sous Programme SBR 1

```
NETWORK16 // Sous programme de remise a jour des tables
75 SBR K1
NETWORK17 // Calcul de Tpulse
78 CALL K10
NETWORK18 // Reactualisation des tables d'activite d'etapes  $T_p = T_s$ 
81 BMW VW20 VW10 K5
NETWORK19 // Initialisation a 0 des tables  $T_a T_d T_{f1} T_{f0}$ 
91 FILL K0 VW40 K30
NETWORK20
101 RET
```

Sous Programme SBR 2

```
NETWORK21 // Calculs preliminaires annexes front ...
102 SBR K2
NETWORK22 // front montant de dcy dans V5.0
105 LD E1.0
107 UN V5.0
```

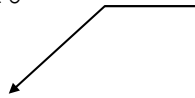


```
110 =          V5.1
NETWORK23
113 LD         E1.0
115 =          V5.0
NETWORK24 // Front de dcy dans V5.0
118 RET
```

Sous Programme SBR 3

```
NETWORK25 // Calcul des receptivites des transitions Table Tt
119 SBR    K3
NETWORK26 // Receptivite T0
122 LDN   E1.3
124 U     V0.0
127 U     V5.1
130 =     V80.0
NETWORK27 // Receptivite T1
133 LD    SM0.0
135 U     V0.0
138 =     V80.1
NETWORK28 // Receptivite T2
141 LD    E1.3
143 U     V0.0
146 =     V80.2
NETWORK29 // Receptivite T3
149 LDN   E1.3
151 U     V0.0
154 =     V80.3
NETWORK30 // Receptivite T10
157 LD    E0.0
159 U     V0.1
162 U     V5.1
165 =     V81.0
NETWORK31 // Receptivite T11
168 LD    SM0.0
170 U     V0.1
173 =     V81.1
NETWORK32 // Receptivite T12
176 LDN   E1.2
178 U     V0.1
181 =     V81.2
NETWORK33 // Receptivite T13
184 LD    E1.1
186 U     V0.1
189 UN    E1.2
191 =     V81.3
NETWORK34 // Receptivite T14
194 LD    SM0.0
196 U     V0.1
```

Calcul des
receptivités et
stockage du
résultat dans la
table Tt





```
199 = V81.4
NETWORK35 // Receptivite T15
202 LDN E1.1
204 U V0.1
207 = V81.5
NETWORK36 // Receptivite T20
210 LD E0.3
212 U V0.2
215 = V82.0
NETWORK37 // Receptivite T21
218 LD E0.1
220 U V0.2
223 = V82.1
NETWORK38 // Receptivite T22
226 LD E0.2
228 U V0.2
231 = V82.2
NETWORK39 // Receptivite T23
234 LD T33
236 U V0.2
239 = V82.3
NETWORK40 // Receptivite T30
242 LD E1.4
244 U V0.3
247 = V83.0
NETWORK41 // Receptivite T31
250 LDN E1.4
252 U V0.3
255 = V83.1
NETWORK42 // Receptivite T32
258 LD E1.5
260 U V0.3
263 UN E1.4
265 = V83.2
NETWORK43 // Receptivite T33
268 LDN E1.5
270 U V0.3
273 = V83.3
NETWORK44 // Fin du sous-programme
276 RET
```

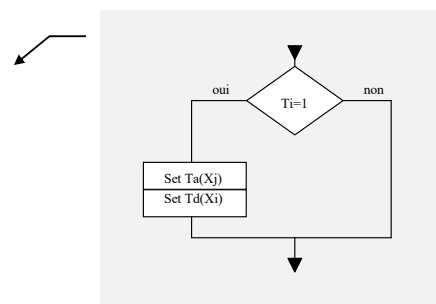
Sous Programme SBR 4

```
NETWORK45 // Calcul de l'evolution du graphe
277 SBR K4
NETWORK46 // Traitement transition par transition (Trans. T0)
280 LD V20.0
283 U V80.0
286 S V40.1 K1
293 S V50.0 K1
```

Si une transition est franchissable, on prépare l'activation des étapes suivantes (via T_a) et la désactivation des étapes précédentes (via T_d)



```
NETWORK47 // Transition T1
  300 LD    V20.1
  303 U    V80.1
  306 S    V40.2      K1
  313 S    V50.1      K1
NETWORK48 // Transition T2
  320 LD    V20.2
  323 U    V80.2
  326 S    V40.3      K1
  333 S    V50.2      K1
NETWORK49 // Transition T3
  340 LD    V20.3
  343 U    V80.3
  346 S    V40.0      K1
  353 S    V50.3      K1
NETWORK50 // Transition T10
  360 LD    V21.0
  363 U    V81.0
  366 S    V41.1      K1
  373 S    V51.0      K1
NETWORK51 // Transition T11
  380 LD    V21.1
  383 U    V81.1
  386 S    V41.2      K1
  393 S    V51.1      K1
NETWORK52 // Transition T12
  400 LD    V21.2
  403 U    V81.2
  406 S    V41.0      K1
  413 S    V51.2      K1
NETWORK53 // Transition T13
  420 LD    V21.0
  423 U    V81.3
  426 S    V41.3      K1
  433 S    V21.0      K1
NETWORK54 // Transition T14
  440 LD    V21.3
  443 U    V81.4
  446 S    V41.4      K1
  453 S    V51.3      K1
NETWORK55 // Transition T15
  460 LD    V21.4
  463 U    V81.4
  466 S    V41.0      K1
  473 S    V51.4      K1
NETWORK56 // Transition T20
  480 LD    V22.0
  483 U    V82.0
  486 S    V42.1      K1
```





```
493 S          V52.0          K1
NETWORK57 // Transition T21
500 LD        V22.1
503 U         V82.1
506 S         V42.2          K1
513 S         V52.1          K1
NETWORK58 // Transition T22
520 LD        V22.2
523 U         V82.2
526 S         V42.3          K1
533 S         V52.2          K1
NETWORK59 // Transition T23
540 LD        V22.3
543 U         V82.3
546 S         V42.0          K1
553 S         V52.3          K1
NETWORK60 // Transition T30
560 LD        V23.0
563 U         V83.0
566 S         V43.1          K1
573 S         V53.0          K1
NETWORK61 // Transition T31
580 LD        V23.1
583 U         V83.1
586 S         V43.0          K1
593 S         V53.1          K1
NETWORK62 // Transition T32
600 LD        V23.0
603 U         V83.2
606 S         V43.2          K1
613 S         V53.0          K1
NETWORK63 // Transition T33
620 LD        V23.2
623 U         V83.3
626 S         V43.0          K1
633 S         V53.2          K1
NETWORK64 // Calcul de  $T_s = (T_p \text{ et (non } T_d) \text{ ) ou } T_a$ 
640 MOVD      VD50           VD2
646 INVD      VD2
650 UNDD      VD10           VD2
656 ORD       VD40           VD2
662 MOVD      VD2            VD20
668 MOVD      VD54           VD2
674 INVD      VD2
678 UNDD      VD14           VD2
684 ORD       VD44           VD2
690 MOVD      VD2            VD24
696 MOVW      VW58           VW2
702 INVW      VW2
```

Calcul global
de tous les
résultats par
paquets de 32
mis 16



```
706 UNDW    VW18      VW2
712 ORW     VW48      VW2
718 MOVW    VW2       VW28
NETWORK65
724 RET
```

Sous Programme SBR5

```
NETWORK66
725 SBR     K5
NETWORK67 // Calcul de Tpulse
728 LD      SM0.0
730 CALL    K10
NETWORK68 // Traitement des forçages etape par etape
733 LD      V20.0
736 S       V71.0      K5
743 S       V72.0      K4
750 S       V83.0      K3
NETWORK69
757 LD      V20.1
760 S       V71.0      K5
767 S       V61.0      K1
NETWORK70
774 LD      V21.0
777 S       V72.0      K4
784 S       V73.0      K3
NETWORK71
791 LD      V21.1
794 S       V72.0      K4
801 S       V62.0      K1
NETWORK72
808 LD      V21.3
811 S       V73.0      K3
818 S       V63.0      K1
NETWORK73 // Calcul du resultat des forçages Ts=(Ts et
           non Tf0) ou Tf1
825 MOVD    VD70      VD2
831 INVD    VD2
835 UNDD    VD20      VD2
841 ORD     VD60      VD2
847 MOVD    VD2       VD20
853 MOVD    VD74      VD2
859 INVD    VD2
863 UNDD    VD24      VD2
869 ORD     VD64      VD2
875 MOVD    VD2       VD24
881 MOVW    VW78      VW2
887 INVW    VW2
891 UNDW    VW28      VW2
```

Vue la méthode employée pour le forçage (2 tables T_{f0} et T_{f1}), Il est possible de traiter les forçages étape par étape comme dans le sous programme N°4



```
897 ORW      VW68      VW2
903 MOVW     VW2       VW28
NETWORK74   // Calcul de Tpulse
909 LD      SM0.0
911 CALL     K10
NETWORK75   // Fin de SRB 5
914 RET
```

Sous Programme SBR 6

```
NETWORK76   // Traitement des tempos et compteurs
915 SBR      K6
NETWORK77
918 LD      V22.2
921 TON     T33      K30
NETWORK78
927 RET
```

Ces actions peuvent
influer sur les sorties
'normales'. Elles sont
donc rassemblées dans
le même sous-
programme et traitées
avant le combinatoire de

Sous Programme SBR 8

```
NETWORK79
928 SBR      K8
NETWORK80   // Mise en place des figeages
931 LDN     V20.3
934 =       V0.1
937 =       V0.2
940 =       V0.3
NETWORK81   // Fonction en combinatoire Voyant C.I.
943 LD      E0.0
945 U       E0.2
947 =       A0.3
NETWORK82
949 LD      V22.2
952 U       E0.0
954 LD      V23.1
957 U       E1.7
959 OLD
960 =       A0.1
NETWORK83
962 LD      V22.0
965 LD      V23.1
968 U       E1.6
970 OLD
971 =       A0.2
NETWORK84
973 LDN     V23.2
976 JMP     K10
NETWORK85
979 LD      V23.2
```



```
982 U      E1.6
984 S      A0.0      K1
NETWORK86
991 LD     V23.2
994 U      E1.7
996 R      A0.0      K1
NETWORK87
1003 JMP   K11
NETWORK88
1006 LBL   K10
NETWORK89
1009 LD     V22.2
1012 U      E0.0
1014 =      A0.0
NETWORK90
1016 LBL   K11
NETWORK91
1019 LD     V21.2
1022 =      A0.4
NETWORK92
1024 LD     V21.4
1027 =      A0.5
NETWORK93
1029 RET
```

Sous Programme SBR 10

```
NETWORK94 // Sous programme de calcul de Tpulse SBR 10
1030 SBR   K10
NETWORK95
1033 MOVD  VD10      VD2
1039 INVD  VD2
1043 UNDD  VD20      VD2
1049 MOVD  VD2       VD30
1055 MOVD  VD14      VD2
1061 INVD  VD2
1065 UNDD  VD24      VD2
1071 MOVD  VD2       VD34
1077 MOVW  VW18      VW2
1083 INVW  VW2
1087 MOVW  VW2       VW38
NETWORK96
1093 RET
```

Ce sous programme limite la taille du code, permet une plus grande souplesse et lisibilité.

$$T_{\text{pulse}} = T_s \cdot \bar{T}_p$$